BAB II
TINJAUAN PUSTAKA

2.1. Teori Dasar

2.1.1. Sistem

Sistem adalah gabungan atau sekelompok dari beberapa komponen yang berisikan proses interaksi antara komponen – kompone lain sehingga menyebabkan sebuah ketergantungan antara komponen satu dengan yang lainnya. Suatu sistem terbentuk untuk mencapai tingkat efektivitas yang signifikan maka hal tersebut dapat membantu proses manajemen yang sedang berjalan (Sutabri, 2012b).

Sistem merupakan sebuah jaringan kerja dari sekelompok prosedur yang berhubungan, kumpulan prosedur yang melakukan kegiatan untuk melakukan suatu perintah tertentu (Hutahaean, 2014).

Dalam prinsipnya sistem dikelompokan menjadi dua bagian yaitu pertama sistem terbuka dan sistem tertutup. Sistem terbuka adalah proses yang berhubungan dengan lingkungan yang berasal dari sumber daya. Sedangkan sistem tertutup merupakan sebuah sistem yang tidak dapat dipengaruhi pihak luar dengan cara mekanisme pengendalian.

2.1.2. Informasi

Informasi adalah sebuah data yang telah diolah menjadi betuk yang memiliki nilai dan berguna untuk menentukan keputusan. Informasi memiliki nilai yang ditentukan oleh dua hal yaitu biaya mendapatkannya dan manfaat (Hutahaean, 2014).

Informasi merupakan masukan yang diperlukan dan penting dalam proses pengambilan keputusan di dalam suatu organisasi. Akan tetapi informasi tidak akan disajikan secara mentah atau utuh, melainkan harus dihasilkan dari data yang sudah diolah (Goyal, 2014).

Berdasarkan dari definisi informasi menurut para ahli di atas, informasi adalah bentuk data yang telah diproses atau diolah sehingga menghasilkan informasi yang dapat membantu proses pengambilan

keputusan. Informasi memiliki nilai untuk memenuhi persyaratan agar dapat menjadi informasi yang tepat untuk mengambil keputusan.

A. Nilai Informasi

Informasi harus memiliki nilai berdasarkan persyaratan yang dibutuhkan manajer untuk pengembilan keputusan. Berikut ini merupakan persyaratan dari informasi (Sutabri, 2012b):

1. Tepat waktu

   Makna dari informasi yang tepat waktu merupakan informasi yang sudah ada dan datang kepada manajer sebelum suatu keputusan diambil. Makna pada kata "tepat" di sini relatif. Suatu informasi yang telah tiba pada manajer satu sehari sebelum pengambilan keputusan mungkin dianggap tepat. Namun belum tentu untuk manajer lain yang lebih sibuk yang menerima informasi seminggu sebelum penentuan keputusan akan dianggap bernilai tepat.

2. Relevan

   Informasi yang disampaikan harus relevan, yakni ada hubungannya dengan kepentingan penerima.

3. Informasi bernilai

   Informasi yang bernilai ialah informasi yang memiliki nilai dan berbobot untuk suatu pengambilan keputusan.

4. *Realiable*

   Informasi yang dapat dipercaya karena hal tersebut sangat berpengaruh pada citra perusahaan atau organisasi. Informasi yang diberikan kepada seseorang ataupun organisasi harus ada kebenerannya.

2.1.3. Sistem Informasi

Sistem informasi adalah sebuah sistem yang ada di suatu organisasi yang berhubungan dengan kebutuhan pengerjaan transaksi harian serta mendukung fungsi operasi perusahaan dengan kegiatan strategi yang bersifat manajerial agar dapat menyajikan laporan – laporan yang diperlukan kepada pihak luar (Sutabri, 2012b).

Sistem informasi merupakan mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan tertentu (Rainer & Prince, 2021).

Sistem informasi adalah suatu proses pengumpulan data, penyimpanan, analisa data atau informasi dengan tujuan tertentu. Sistem informasi yang berupa data *input* dan akan menghasilkan sebuah laporan atau *output* sehingga hasil tersebut dapat diterima oleh sistem lainnya dan kegiatan pada organisasi atau perusahaan dalam menentukan keputusan (Prehanto, 2020).

Berdasarkan dari penjelasan tentang sistem informasi dari beberapa sumber kutipan, maka sistem informasi adalah suatu proses sistem yang mengelola data atau informasi yang telah ter-*input* dan menghasilkan sebuah *output* berupa informasi baru yang lebih bernilai hingga dapat membatu pengguna untuk menentukan keputusan.

A. Karakteristik Sistem Informasi

Suatu sistem harus memiliki beberapa karakteristik. Hal tersebut akan dinilai dan membuat sebuah sistem berbeda dari sistem yang lainnya. Hadirnya sistem informasi sebagai alat bantu untuk meminimalisir kesalahan atau *human error* dan sebagai bahan pengambil keputusan. Untuk mengetahui tentang karakteristik sistem informasi berikut di bawah ini merupakan penjelasannya (Andrianof, 2018):

1. *Component*

   Sistem harus terdapat kumpulan komponen yang diantaranya adanya interaksi sehingga membentuk satu kesatuan dan bekerja sama dengan berbagai cabang sistem.

2. Lingkungan luar sistem (*Environment*)

   Lingkungan luar sistem sangat berpengaruh karena memiliki sifat menguntungkan yang harus dijaga dan merugikan tetap dijaga namun harus dapat dikendalikan.

3. Batasan sistem (*Boundary*)

Ruang lingkup atau *scope* adalah batasan dari daru lingkup luar sistem atau dapat dikatakan batas sistem yang sesuai dengan bundaran daerahnya.

4. Penghubung sistem (*Interface*)

   Penghubung sistem adalah sebuah alat bantu yang digunakan untuk mengubungkan satu subsistem dengan subsistem lainnya. Dengan penghubung sistem maka sumber – sumber daya akan mengalir dan mengeluarkan hasil (*output*) dari subsistem dan akan menjadi masukkan (*input*) untuk subsistem lainnya.

5. Masukkan sistem (*input*)

   Masukkan sistem adalah sebuah sumber daya atau data yang dimasukkan kedalam aplikasi atau sistem, sumber data atau data tersebut dapat berupa *maintenance input* dan *signal input. Maintenance input* adalah sumber daya yang di *input* agar pada sistem dapat berjalan. Sedangkan *signal input* merupakan sumber daya yang diproses untuk mendapatkan mengeluarkan *output.*

6. Keluaran sistem (*output*)

   Keluaran sistem merupakan suatu energi yang dikeluarkan setelah proses *input* selesai.

7. Pengolah sistem

   Pengolah sistem adalah proses yang mengubah *input* menjadi bentuk *output*.

8. Sasaran sistem

   Sasaran sistem adalah tujuan atau *goal* dari suatu proses yang terjadi.

B. Tujuan dan Manfaat Sistem Informasi

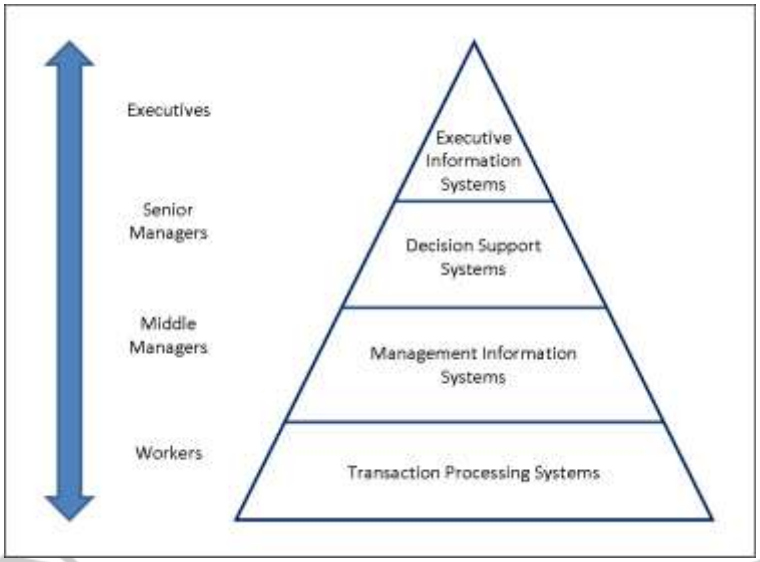
Tujuan dirancangnya sebuah sistem adalah untuk menghasilkan sebuah produk yang tepat untuk para penggunanya. Produk tersebut antara lain yaitu pesan, laporan berupa formulir dan grafik, tampilan video, audio, produk cetakan kertas, hingg multimedia (Samsudin et al., 2019).

Sistem informasi memiliki beberapa manfaat untuk penggunanya diantaranya, yaitu (Jamaludin, 2019):

1. Meningkatkan ketersediaan data yang akan diberikan secara tepat waktu dan akurat untuk pemakainya.
2. Menjamin kualitas dan kepandaian dalam memanfaatkan suatu sistem informasi dengan kritis.
3. Mengelaborasikan proses perencanaan secara efektif.
4. Menganalisis kebutuhan terkait dengan pendukung sistem informasi.
5. Mengantisipasi konsekuensi ekonomis yang disebabkan oleh sistem informasi.
6. Memperbaiki proses bisnis yang terjadi pada aplikasi pengembangan dan pemeliharan sistem (*maintenance*).
7. Mengolah transaksi yang terjadi, mengurangi pengeluaran dan biaya, hingga dapat menghasilkan pendapatan dari produk atau pelayanan organisasi.

C. Tipe Sistem Informasi

Sistem informasi (SI) merupakan gabungan dari teknologi informasi dan kegiatan penggunannya untuk mendukung aktivitas operasional dan manajemen. Istilah dari sistem informasi sering merujuk pada interaksi antara proses algoritmik, data, orang (*user*), serta teknologi. Istilah tersebut digunakan tidak hanya untuk penggunaan organisasi teknologi informasi (TI), namun digunakan oleh *user* untuk berinteraksi dengan teknologi. Berikut ini merupakan tipe – tipe pada sistem informasi yang dapat dilihat pada **Gambar 2.1.** di bawah ini.



Gambar 2.1. Tipe Sistem Informasi

Sumber: (upanastudio.com)

Pada **Gambar 2.1.** di atas merupakan piramida tipe – tipe sistem informasi, berikut merupakan penjelasan dari tipe sistem informasi (Simarmata et al., 2020):

1. *Transaction Processing Systems* (Sistem Operasional)

   Sistem operasional merupakan tingkatan sistem operasional yang berada di bagian bawah piramida. Operasional ini biasanya dilakukan oleh pekerja lantai atau *staff* perusahaan yang memegang atau menyediakan data yang diperlukan pada manajemen operasional. Data tersebut didapatkan dengan cara pelacakan otomatis dari aktivitas di tingkat rendah dan transaksi dasar.

2. *Management Information Systems* (Sistem Informasi Manajemen)

   Sistem Informasi Manajemen merupakan sistem di tingkat manajemen yang dipakai oleh manajer untuk mendukung dan memastikan proses organisasi dalam jangka pendek dan menengah dinilai lancar. Data informasi yang telah disediakan sistem dapat menguatkan seorang manajer untuk melakukan evaluasi pada setiap kinerja organisasi dengan melihat dan membandingkan data keluaran saat ini dengan sebelumnya. Informasi yang dihasilkan akan bernilai untuk manajemen mau itu di *level* teratas, menengah, dan dasar.

3. *Decision Support Systems* (Sistem Pendukung Keputusan)

   Decision Support Systems (DSS) merupakan sistem berbasis pengetahuan yang digunakan oleh seorang manajer senior yang menyediakan fasilitas untuk penciptaan pengetahuan dan menguatkan integrasi ke organisasi. Sistem pendukung keputusan digunakan untuk melakukan analisis informasi terstruktur sehingga manajer dapat memproyeksikan dampak potensial dari keputusan ke masa yang akan datang. Sistem ini biasanya digunakan dan interaktif untuk memecahkan suatu masalah. Fasilitas yang diberikan yaitu dapat mengakses ke database, alat analitis, dan mendukung pertukaran data informasi pada suatu organisasi.

4. *Executive Information Systems* (Sistem Informasi Ekslusif)

   Executive Information Systems (EIS) adalah sistem yang berada di tingkat atas piramida. Sistem ini dapat membantu seorang eksekutif dan manajer senior untuk melakukan analisis lingkungan perusahaan atau organisasi beroperasi. Analisis tersebut dilakukan untuk mengidentifikasi tren jangka panjang dan melakukan perencanaan tindakan apa yang sesuai. Namun informasi pada sistem seperti ini sering ditemukan terstruktur dengan lemah dan informasi tersebut didapatkan dari sumber internal dan eksternal. Sistem ini dirancang digunakan langsung oleh seorang eksekutif tanpa adanya perantara dan mudah untuk disesuaikan dengan preferensi penggunanya.

Berdasarkan penjelasan tentang tipe sistem informasi di atas, penelitian ini termasuk tipe sistem informasi *Transaction Processing Systems.* Tipe sistem tersebut memiliki kesamaan dalam melakukan manajemen aset dan pendataan reservasi *meeting room* dan alat transportasi perusahaan.

2.1.4. Manajemen Aset

Manajemen aset merupakan upaya organisasi untuk merencanakan, melaksanakan, meninjau, dan meningkatkan proses dan kegiatan pengelolaan aset guna memaksimalkan pencapaian nilai dari aset. Maka dari

itu, pengelolaam aset tidak berfokus pada aset itu sendiri, tetapi pada nilai yang dapat diperoleh dari organisasi (Souza, 2021).

Pada manajemen aset di dalamnya mencakup proses perencanaan (*planning*), perancangan (*designing*), pengorganisasian (*organization*), penggunaan (*utilization*), pemeliharaan (*maintenance*), penghapusan (*removal*), hingga pengawasan (*supervision*) pada aset. Proses tersebut dilakukan secara terstruktur dan sistematis pada siklus hidup aset (Wahyuni & Khoirudin, 2020).

Dari pemaparan pengertian tentang manajemen aset menurut para ahli di atas, manajemen aset adalah suatu kegiatan yang dilakukan oleh perusahaan secara terstruktur agar tercapainya nilai suatu aset. Proses untuk melaksanakan manajemen aset mencangkup beberapa proses yaitu perencanaan, perancangan, pengorganisasian, penggunaan, pemeliharaan, penghapusan, serta pengawasan.

A. Asset Life Cycle Management

Asset Life Cycle Management adalah sebuah pendekatan yang digunakan untuk proses pengelolaan suatu aset berdasarkan siklus hidup aset (Macchi et al., 2018). Terdapat beberapa tahapan pada siklus manajemen aset, tahap – tahap tersebut dapat membantu perusahaan dalam proses pengelolaan keuangan dan aset dengan efisien. Berikut ini merupakan siklus manajemen aset yang dapat dilihat pada **Gambar 2.2.** di bawah ini.

Gambar 2.2. ***Asset Life Cycle Management***

Sumber: (infraon.io)

Pada gambar di atas merupakan tahapan pada siklus manajemen aset. Berikut ini merupakan penjelasan dari tahapan – tahapan siklus manajemen aset (Silha & Setyowati, 2019):

1. *Planning & Acquisition* (Perencanaan & Perolehan)

   Planning merupakan tahapan penyusunan rencana strategi terkait dengan aset yang dilakukan oleh suatu organisasi. Sedangkan *Acquisition* adalah suatu tahapan kegiatan untuk mendapatkan barang ataupun jasa yang sudah direncanakan.

2. *Utilization* (Pemanfaatan)

   Utilization merupakan tahapan yang di mana organisasi akan menilai aset yang ada. Hal tersebut dapat menentukan rincian nilai kekayaan yang telah dimiliki serta catatan tentang aset yang sudah dialihkan hingga dihapus.

3. *Operation & Maintenance* (Pengoperasian & Pemeliharaan)

   Operation adalah tahapan dimana anggota menggunakaan dan memanfaatkan aset untuk keperluan tugas ataupun pekerjaan untuk mencapai suatu tujuan. Sedangkan *Maintenance* merupakan tahapan yang dilakukan untuk memperbaiki ataupun memperbarui agar dapat digunakan dalam jangka Panjang.

4. *Disposal or Renewal* (Penghapusan atau Pembaharuan)

   Disposal or renewal adalah tahapan jika suatu aset tidak dapat digunakan kembali maka akan dilakukan pembuangan atau penghapusan aset. Aset yang sudah dihapus tersebut nantinya akan dilakukan penggantian aset yang baru agar kinerja dari karyawan dapat tetap stabil.

B. Tujuan dan Nilai Manajemen Aset

Definisi manajemen aset adalah kegiatan yang terstrukur oleh suatu organisasi dengan tujuan mewujudkan nilai asetnya. Nilai aset pada manajemen aset bisa berbeda – beda pada setiap organisasi dan *stakeholder*. Maka dari itu munculnya perdebatan terkait dengan definisi nilai pada suatu aset akan sangat mungkin terjadi. Perusahaan

yang menerapkan manajemen asset dapat memperoleh nilai berupa (Cahyo, 2020):

1. Memperbaiki kinerja keuangan, dengan meningkatkan pengembalian investasi dan mengurangi dana pengeluaran, sehingga dapat mempertahankan nilai aset tanpa harus mengorbankan realisasi jangka pendek maupun jangka Panjang tujuan perusahaan.
2. Terkoordinasi dan terukurnya keputusan mengenai investasi aset, memungkinkan perusahaan untuk meningkatkan pengambilan keputusan secara efektif dengan menyeimbangkan biaya, resiko, peluang, dan kinerja.
3. Resiko terhadap aset dapat lebih diatur dengan baik. Dengan mengurangi kerugian finansial perusahaan, meningkatkan kesehatan serta keselamatan, kepastian yang baik dan reputasi, dapat meminimalir dampak dari lingkungan dan sosial, sehingga berdampak pada berkurangnya kewajiban seperti premi asuransi, denda, dan hukuman.
4. Meningkatkan hasil atau *output* dan memperbaiki pelayanan. Suatu perusahaan yang telah menjalankan manajemen aset akan menjamin kinerja aset dapat memberikan peningkatan layanan dan produk secara konsisten terpenuhi atau melebihi dari harapan pelanggan dan *stakeholder*.
5. Berperan di dalam *social responsibility*, perusahaan secara tidak langsung telah bertanggung jawab dalam berperan pada pembangunan ekonomi untuk meningkatkan kualitas kehidupan.
6. Mempertunjukkan kepatuhan terhadap hukum, secara transparan telah sesuai dengan persyaratan hukum, undang – undang, dan peraturan yang berlaku, serta menjunjung standar manajemen aset, kebijakan dan proses.
7. Repurtasi perusahaan yang lebih baik, dengan meningkatkan kepuasan pada pelanggan, dan kesadaran dan kepercayaan terhadap *stakeholder*.

8. Mempertahankan perusahaan, mengelola dengan efektif pada efek jangka pendek dan panjang, kinerja dan pengeluaran, dapat meningkatkan keberlangsungan perusahaan.
9. Meningkatkan efisiensi dan efektivitas, melakukan peninjauan dan meningkatkan proses, prosedur dan kinerja aset sehingga dapat meningkatkan efisiensi dan efektivitas, serta dapat mencapai tujuan perusahaan.

Proses untuk merealisasikan nilai dari aset di suatu perusahaan akan mengikutsertakan biaya, resiko, kesempatan, dan kinerja dari aset untuk dilakukan penyeimbangan (Cahyo, 2019).

C. Manfaat Manajemen Aset

Manajemen aset dikatakan baik jika dapat memberikan manfaat pada suatu organisasi atau perusahaan secara efektif dan efisien memberikan pengetahuan bisnis, dan mencapai tujuannya dalam mendapatkan keuntungan dan pelayanan. Berikut ini merupakan beberapa manfaat dari manajemen aset (Putra, 2019):

1. Proses secara sistematis pada setiap keputusan yang beruhubungan dengan aset sehingga tujuan bisnis seperti persyaratan aset, perolehan, serta pembuangan dapat sesuai.
2. Meningkatkan kinerja aset karena adanya tunjangan atau dukungan logistik.
3. Efektif dalam proses internal untuk mengelola aset.
4. Dapat memenuhi target bisnis dan peraturan seperti target operasional, perturan lingkungan, target keuangan, peraturan kesehatan dan keselamatan, manajemen resiko, dan syarat asuransi.

2.1.5. *User Interface*

User interface (UI) merupakan kumpulan komponen perangkat lunak maupun perangkat keras yang dapat digunakan pengguna untuk mengkontrol suatu tugas dengan menggunakan sistem. Komponen yang dimiliki UI berfungsi untuk penerjemah sekaligus penghubung informasi yang berasal dari sistem kepada pengguna dan sebaliknya (Solichuddin, 2021).

A. Prinsip Perancangan UI

Dalam merancang sebuah *user interface* perlu memenuhi persyaratan agar dapat dinilai baik. Terdapat delapan prinsip yang mendasari perancangan UI, yaitu (Solichuddin, 2021):

1. Prinsip Struktur

   Desain UI wajib dirancang sesuai dengan kegunaannya berdasarkan komponen yang mudah dikenali, salah satu contohnya seperti penggunaan logo atau *icon* pada tombol.

2. Prinsip Kesederhanaan

   Dalam merancang sebuah desain UI harus dibangun sesederhana mungkin. Kesederhanaan tersebut dapat terlihat dari tampilan keseluruhan desain UI hingga alur dari hasil desain.

3. Prinsip Visibilitas

   Desain yang tidak membuat penggunanya keliru dengan pilihan yang tidak seharusnya ada, merupakan sebuah desain UI baik. Salah satu contoh dari prinsip ini adalah menyediakan menu yang jarang pengguna pakai.

4. Prinsip *Feedback*

   Pengguna harus mendapatkan suatu peringatan yang berkaitan dengan perubahan aksi, kondisi, maupun kesalahan yang telah dilakukan oleh pengguna. Contohnya adalah sebuah pemberitahuan jika pengguna telah merubah data.

5. Prinsip Toleransi

   Meminimalisir kesalahan yang akan dibuat oleh pengguna dengan membuat desain menjadi sefleksibel mungkin. Prinsip ini mendorong perancang agar menyediakan informasi yang diperlukan dan seringkas mungkin.

6. Prinsip *Reuse*

   Komponen dari UI harus dapat digunakan ulang dengan sesuai tujuan. Contoh dari prinsip ini ialah perancang harus membuat desain yang sesuai dengan kebutuhan dari pengguna agar komponen dapat digunakan dengan baik.

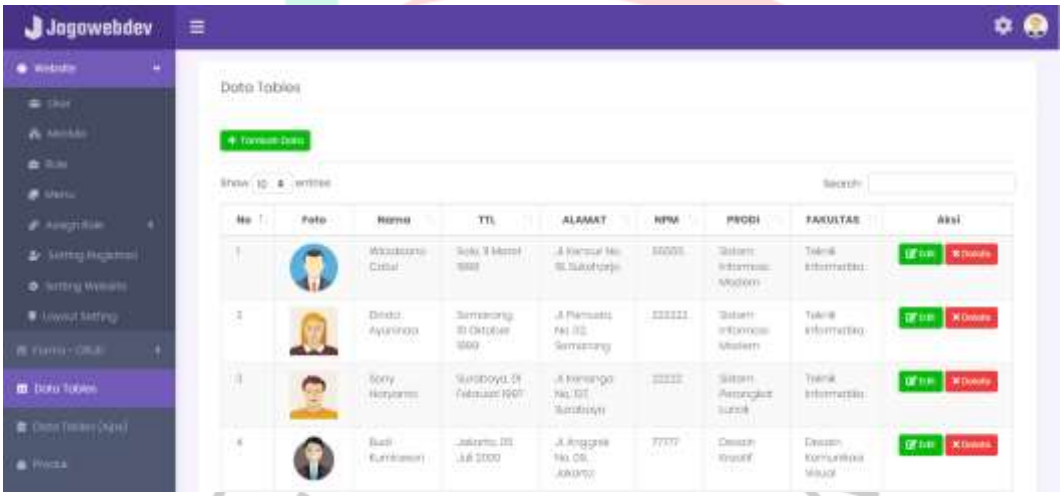
7. Menjaga Perhatian

   Desain UI dapat dikatakan baik adalah dapat membuat pengguna tertarik dan dapat mempertahankannya. Dalam mempertahankan ketertarikan pengguna, perancang harus memperhatikan komponen dari UI, seperti pemilihan warna hingga penempatan konten atau informasi untuk pengguna.

8. Bahasa yang sederhana

   Bahasa yang dipilih pada pembuat desain UI lebih baik menggunakan bahasa umum yang banyak diketahui oleh semua pengguna.

Berdasarkan penjelasan mengenai *user interface* di atas berikut ini merupakan contoh desain UI yang dapat dilihat pada **Gambar 2.3.** di bawah ini.



Gambar 2.3. Contoh *User Interface*

Sumber: (www. jagowebdev.com)

Pada **Gambar 2.3.** di atas merupakan salah satu contoh desain *user interface* halaman data mahasiswa. Berdasarkan contoh desain telah mengimplementasikan prinsip – prinsip perancangan *user interface*, seperti penggunaan kata yang sederhana, pemilihan warna yang nyaman dilihat oleh pengguna, terdapat simbol *edit*, *delete*, dan tambah, serta menampilkan informasi yang diperlukan.

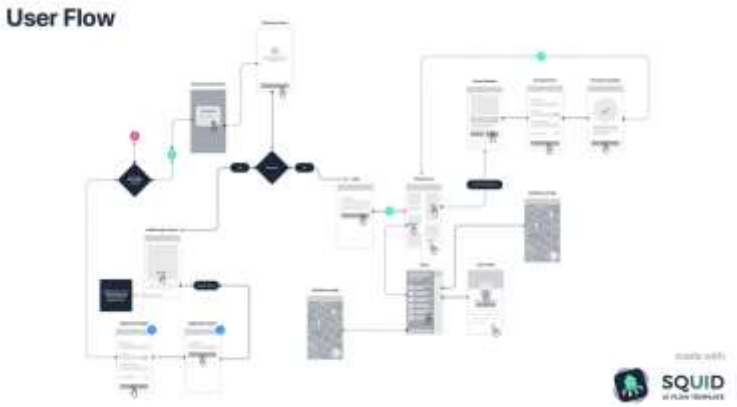
2.1.6. *User Flow*

User flow adalah sebuah diagram alur atau tahapan – tahapan pengguna yang harus dilakukan untuk mencapai suatu aktivitas. Desain dari *user flow* harus sesuai dengan proses bisnis perusahaan ataupun kebiasaan penggunanya. Konversi kebiasaan pengguna ke dalam sebuah bentuk digital berupa diagram atau *flow* yang telah didapatkan bedasarkan hasil analisa atau wawancara. Tahap *user flow* dibentuk agar dapat menentukan desain yang tepat untuk perancangan UI (*User Interface*)(Santoso, 2022).

Tabel 2.1. Komponen User Flow

Simbol	Nama
	*Interaction*
	*Decision node*
	*Start / End*

Berdasarkan penjelasan konsep dasar dan komponen dari *user flow* berikut ini merupakan contoh dari *user flow* yang dapat dilihat pada **Gambar 2.3.** di bawah ini.



Gambar 2.4. Contoh *User Flow*

Sumber: (fahrizalbimantara.medium.com)

Pada **Gambar 2.4.** di atas adalah contoh *user flow e-commerce* yang hampir sesuai dengan penjelasan dari konsep di atas. Pada *user flow* terdapat komponen interaksi, *decision* atau pilihan, dan titik mulainya sebuah *flow*. *User flow* tersebut menggambarkan sebuah alur pengguna mulai dari login, pemilihan produk, hingga *check out* produk.

2.1.7. Database

Database adalah repositori atau penyimpanan yang berisi banyak berkas individu. Namun ini bukan sekedar repostori dan dipandang lebih luas. Sebuah *item* data biasanya tidak ada dalam isolasi dalam database. Namun *item* data terkait satu sama lain. Maka database menyimpan kumpulan *item* data yang saling terkait satu sama lain serta informasi dari relasinya (Haldar, 2015).

Saat ini penggunaan dari sistem informasi dan database sudah menjadi khalayak umum karena database merupakan salah satu komponen untuk menyajikan bermacam informasi. Dengan demikian penggunaan database di dalam perancangan sebuah sistem yang harus diperhatikan dengan baik, salah satunya pada pemilihan *Database Management System* (DBMS). DBMS adalah sebuah *software* yang gunakan untuk mengendalikan pembuatan, pengolahan, pemeliharaan, hingga pemakaian data dengan jumlah yang besar (Warman & Ramdaniansyah, 2018).

2.1.8. *Entity Relationship Diagram* (ERD)

Entity relationship diagram (ERD) merupakan sebuah penggambaran model yang berbasis pada persepsi yang nyata. Pemodelan ERD terdiri dari objek – objek yaitu *entity* atau entitas dan relasi atau hubungan antar objek. Dalam mengambarkan sebuah ERD ada baiknya menentukan tingkat derajat pada setiap entitas. Derajat entitas merupakan ukuran yang memiliki sifat yang kompleks pada relasi data. Sedangkan entitas merupakan objek yang dapat dibedakan dari objek lainnya pada oleh atribut. Sebagai contoh sebuah entitas produk berbeda dengan entitas pelanggan. Hal tersebut dikarenakan dua entitas memiliki atribut yang berbeda (Lubis, 2016).

Pada setiap entitas yang memiliki hubungan dengan entitas lain wajib mempunyai batasan atribut minimal hingga maksimal. Pembatasan minimal dan maksimal yang terdapat pada entitas biasa disebut dengan *cardinality* atau kardinalitas. Kardinalitas menjelaskan minimal dan maksimal pada suatu kejadian entitas yang berpotensi memiliki relasi pada entitas lainnya. Pada kardinalitas digambarkan menjadi beberapa pembatas, yaitu (Lubis, 2016):

1. *One-to-one* (1-1)

   Kardinalitas *one-to-one* menjelaskan entitas memiliki hubungan paling banyak satu entitas.

2. *One-to-many* (1-N)

   Pada kardinalitas *one-to-many* menjelaskan bahwa pada artibut entitas memiliki hubungan pada satu atau banyak artibut entitas lainya.

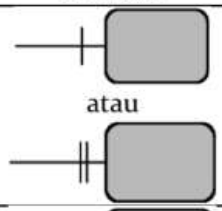
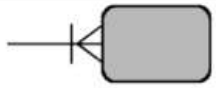
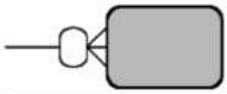
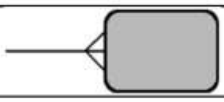
3. *Many-to-one* (N-1)

   Pada *many-to-one* merupakan kebalikan dari *one-to-many*, kardinalitas ini menjelaskan lebih dari satu entitas memiliki hubungan dengan satu entitas.

4. *Many-to-many*

   Kardinalitas *many-to-many* menjelaskan bahwa banyak entitas memiliki relasi dengan banyak entitas lainnya.

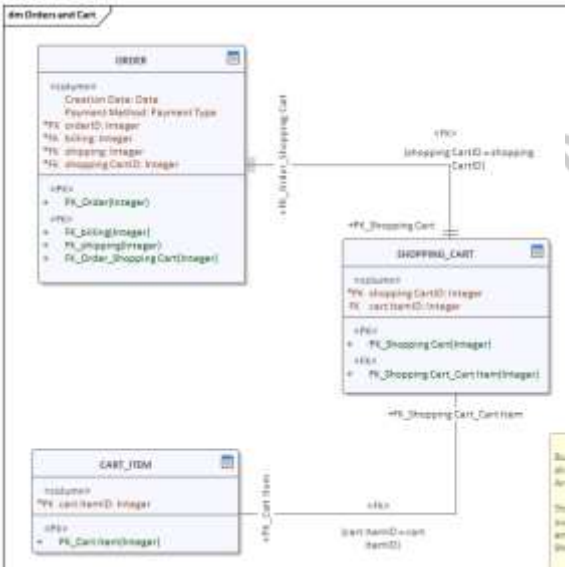
Kardinalitas ERD memiliki beberapa notasi. Berikut ini merupakan notasi kardinalitas ERD yang dapat dilihat pada **Gambar 2.5.** di bawah ini.

Interpretasi Kadinal	Kejadian Minimum	Kejadian Maksimum	Notasi Gambar
Satu dan hanya satu (1,1)	1	1	
Kosong atau satu (0,1)	0	1	
Satu atau lebih (1,N)	1	Banyak / Many (> 1)	
Kosong, satu, atau lebih (0,N)	0	Banyak / Many (> 1)	
Lebih dari satu (N,N)	> 1	> 1	

Gambar 2.5. Notasi Kardinalitas ERD

Sumber: (Lubis, 2016)

Pada **Gambar 2.5.** di atas merupakan notasi kardinalitas ERD. Berdasarkan penjelasan dari konsep ERD dan notasi kardinalitas, berikut ini merupakan contoh ERD yang dapat dilihat pada **Gambar 2.6.** di bawah ini.



Gambar 2.6. Contoh *Entity Relationship Diagram*

Sumber: (www.sparxsystems.com)

Pada **Gambar 2.6.** di atas merupakan contoh dari ERD *orders and cart*. Berdasarkan contoh di atas terdapat tiga entitas data yaitu *order, shopping_cart,* dan *cart_item.* Tabel tersebut memiliki relasi atau hubungan melalui *primary key* yang disimbolkan dengan PK sedangkan *foreign key* disimbolkan dengan FK. Pada entitas *shopping_cart* sebagai *parent* memiliki notasi kardinalitas *one to one* dengan entitas *order* sebagai *child.* Sedangkan entitas *cart_item* sebagai *parent* terdiri dari entitas *shopping_cart* sebagai *child.*

2.1.9. SDLC (*System Development Life Cycle*)

Dalam merancang sebuah sistem informasi dibutuhkan sebuah metode pengembangan sistem agar pada saat melakukan perancangan akan lebih terstruktur. Metode SDLC (*Sytem Development Life Cycle*) adalah salah satu metode klasik pengembangan sistem yang banyak digunakan oleh seorang *analyst* untuk pengembangan, pemeliharaan, dan penggunaan sistem. Metode SDLC adalah sebuah proses untuk mengembangkan atau melakukan perubahan pada sistem dengan menggunakan metodologi yang sudah banyak digunakan oleh seorang *analyst* untuk mengembangkan sistem atau *software* baru ataupun sebelumnya yang berdasarkan *best practice* (Firmansyah & Udi, 2017).

Setiap langkah atau tahap pada metode SDLC memilik tujuan tertentu untuk mendukung penyusunan sistem secara efisien dan efektif. Berikut ini merupakan tahap – tahap pada metode SDLC yaitu (Abdullah, 2017):

1. Perencanaan

   Tahapan ini merupakan studi awal dalam membangun suatu sistem baru.
2. Analisis

   Pada tahap ini merupakan proses identifikasi dan memahami sistem telah ada dengan tujuan untuk merancang sebuah sistem baru atau mengembangkan sistem yang sudah ada.

3. Perancangan atau Desain

   Tahap ini merupakan tahap untuk mendesain serta merancang sebuah sistem baru dengan tujuan agar dapat berjalan lebih maksimal dan dapat mengatasi permasalahan yang sedang terjadi serta dapat mengantisipasi kemungkinan – kemungkinan yang akan terjadi di masa yang akan datang.

4. Penerapan

   Penerapan atau *implementation* merupakan tahapan untuk memperoleh dan koseptual, dan mengidentifikasi sumber daya agar dapat menghasilkan sistem yang bekerja.

5. Pemeliharaan

   Tahap pemeliharaan atau biasa disebut dengan *maintenance* adalah suatu aktifitas untuk melakukan perbaikan *bug* dan *error* pada sistem yang telah selesai dibuat.

Pada metode SDLC terdapat empat model pendekatan dalam siklus hidup sistem, yaitu *Waterfall Model*, *Agile*, *Spiral*, serta Iteratif dan Inkremental. Pada penelitian ini menggunakan model pendekatan *Agile.*

2.1.10. Metodologi *Agile*

Metodologi *Agile* merupakan pendekatan pada siklus hidup sistem atau perangkat lunak (SDLC). Pada metodologi ini lebih menekankan fleksibilitas, adaptasi, serta kerjasama tim *developer* dengan *user*. Dalam pengembangan sistem dilakukan dengan *iterative* dan *incremental*. Hal tersebut bertujuan untuk mendapatkan umpan balik sehingga dapat mempengaruhi pengembangan sistem (Zien et al., 2023).

Pada metodologi *Agile* memiliki kelebihan yaitu adaptasi yang tinggi dan fleksibilitas. Hal tersebut akan membuat *developer* dapat menanggapi perubahan – perubahan dari kebutuhan pengguna secara cepat dan efektif. Tetapi, pada metodologi ini terdapat kelemahan yaitu perencanaan yang lemah dan dokumentasi yang terlalu formal. Hal tersebut menyebabkan *devepoler* menjadi tidak terstruktur dan sulit diukur (Zien et al., 2023).

2.1.11. *Object-Oriented Analysis and Design (OOAD)*

Konsep *Object Oriented Analysis and Design* (OOAD) meliputi analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD) pada sistem dengan menggunakan pendekatan *object.* OOA merupakan metode Analisa yang mengidentifikasi *requirement* pada sistem dari pandangan kelas – kelas serta objek – objek yang terdapat pada lingkup organisasi. OOD merupakan metode yang digunakan untuk mengatur serta memberikan arah dalam arsitektur sistem yang didasarkan oleh manipulasi objek sistem ataupun subsistem (Setiawan, 2019).

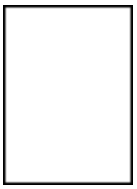
Pada saat melakukan perancangan OOAD memerluka alat bantu yaitu *Unified Modeling Language* (UML). Menurut Henderi (2020) pada (Rahardja & Efana Rahwanto, 2020) UML merupakan suatu *tools* yang membantu dalam pengembangan sistem berbasis objek yang dinilai sangat handal. Fungsi dari UML sendiri yaitu sebagai penghubung dari beberapa aspek pada sistem melalui sebagian elemen grafis yang dikombinasikan menjadi diagram. Pada UML terdapat 14 jenis diagram namun untuk melakukan implementasi hanya menggunakan empat jenis diagram yang sering digunakan, yaitu *Use Case Diagram, Activity Diagram, Class Diagram,* dan *Sequence Diagram*.

2.1.12. *Use Case Diagram*

Use Case Diagram adalah notasi pemodelan yang dirancang paling awal dalam sebuah pengembangan sistem. *Use case* akan mengidentifikasi bagaimana suatu sistem usulan akan digunakan dan mengidentifikasi kategori pengguna yang dapat mengakses atau berinteraksi pada sistem (Rahardja & Efana Rahwanto, 2020). Berikut ini merupakn komponen dari *use case diagram* yang dapat dilihat pada **Tabel 2.2.** (Destriana et al., 2021) di bawah ini.

Tabel 2.2. Komponen *Use Case Diagram*

Simbol	Nama	Keterangan
	Aktor	Aktor merupakan pemodelan jenis peran yang berinteraksi dengan *use case* yang terkait.
	*Use Case*	*Use case* adalah beberapa kumpulan dari perilaku yang dilakukan oleh sistem. *Use case* akan memberikan *output* yang dapat digunakan oleh satu atau lebih aktor.
	Association	*Association* merupakan penghubung antara satu objek dengan objek lainnya.
	Include	*Include* adalah sebuah relasi antar dua *use case* yang menjelaskan *use case* tambahan ke suatu *use case* lainnya untuk menjalakan prosesnya.
	Extend	Menjelaskan bahwa *use case* diperluas sehingga fungsional dari *use case* menambah.

	*Boundaries System*	*Boundaries system* adalah sebuah batas dari sistem.

High level use case merupakan sebuah ringkasan dari tugas aktor yang ditulis dengan tidak terstruktur panjang. Tujuan dari *high level use case diagram* ialah dapat memberikan detail tugas untuk pengguna tentang gambaran dari kompleksitasnya dan mempermudah pengembang untuk melakukan pengelompokkan tugas pengguna terkait dengan sistem yang sedang dikembangkan (Aquino et al., 2020).

A. Karakteristik Use Case

Use case menjelaskan tentang kebutuhan sistem dari pandangan pengguna, yang dapat menciptakan sebuah urutan peristiwa yang berfokuskan pada tujuan yang dapat diikuti oleh pengguna dan *developer* menjadi mudah. Sebuah *use case* yang lengkap di dalamnya mencangkup aliran utama atau dasar dan kumpulan aliran alternatif. Aliran alternatif atau *alternative flow* disebut dengan *use case* ekstensi yang menggambarkan varian aliran normal ke aliran dasar dengan disertai situasi yang tidak biasa (Kurniawan, 2018). *Use case* memiliki beberapa karakteristik berikut diantaranya (Kurniawan, 2018):

- Mengatur *functional requirement.*
- Memodelkan suatu tujuan dari interaksi sistem dan aktor.
- Mencatat bagian dari sebuah peristiwa pemicu ke tujuan.
- Menggambarkan satu aliran utama dan berbagai aliran alternatif.
- Satu penghubung dapat menggunakan fungsionalitas dari penghubung lainnya.

B. Jenis Use Case

Use case memiliki dua jenis yang berbeda, yaitu *business use case* dan *system use case.* Berikut ini penjelasan dari kedua jenis *use case*:

- *Business Use Case*

  Use case bisnis merupakan sebuah gambaran abstrak yang ditulis dengan secara teknologi agnostic dan hanya merujuk pada sebuah proses bisnis yang dijelaskan serta aktor yang terlibat didalamnya. *Use case* bisnis dapat mengidentifikasi sebuah urutan tindakan yang perlu dilakukan oleh bisnis agar dapat memberikan hasil yang bernilai dan dapat digunakan hingga diamati oleh pengguna (Kurniawan, 2018).

- *System Use Case*

  Sedangkan *system use case* ditulis dengan lebih detail dan banyak dibandingkan *business use case* dan lebih mengacu pada sebuah proses yang lebih spesifik pada berbagai bagian yang akan terjadi pada sistem untuk mencapai tujuan akhir. Sebuah *system use case* yang mendetail kepada spesifikasi fungsional, termasuk ketergantungan, fitur pendukung internal yang dibutuhkan hingga fitur internal yang bersifat opsional (Rahardja & Efana Rahwanto, 2020).

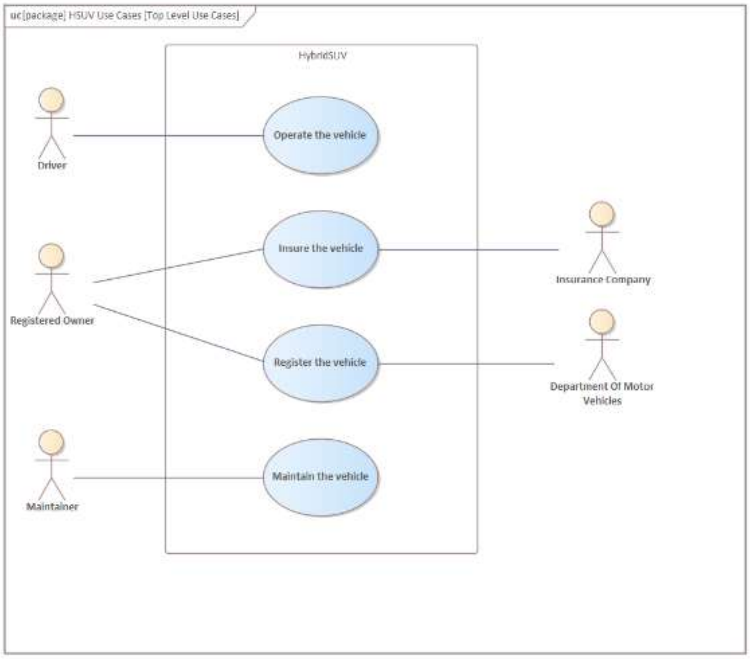
C. Manfaat Use Case dan Contoh Use Case Diagram

Sebuah *use case* dapat menguntungkan *developer* dengan menggambarkan bagaimana sistem harus berperilaku serta mengidentifikasi kesalahan yang akan terjadi pada prosesnya. Berikut ini merupakan manfaat dari penggunaan *use case* (Habibi & Suryansah, 2020).

1. Daftar dari beberapa tujuan yang dibuat pada proses penulisan dapat digunakan untuk menentukan kompleksitas dan biaya sistem.
2. Penulisan berfokus pada pengguna dan sistem maka kebutuhan sistem nyata dapat mengidentifikasi lebih awal dalam proses desain.

3. Karena sebuah *use case* umumnya ditulis dalam Bahasa naratif maka dapat dipahami oleh para *stakeholder*, pengguna, dan eksekutif.
4. Pembuatan perpanjang *use case* dan identifikasi pengecualian untuk scenario *use case* dapat menghemat waktu pengembangan dengan membuanya menjadi lebih mudah untuk mendefinisikan *system requirement.*
5. Desain yang sudah jadi sebelum waktunya dapat dihindari dengan lebih berfokus pada mengidentifikasi apa yang harus dilakukan pada sistem daripada bagaimana seharusnya sistem melakukannya.

Berdasarkan penjelasan tentang konsep dasar dari *use case* di atas. Berikut ini merupakan contoh *use case* yang dapat dilihat pada **Gambar 2.7.** di bawah ini.



Gambar 2.7. Contoh *Use Case*

Sumber: (www.sparxsystems.com)

Pada **Gambar 2.7.** di atas merupakan contoh dari *use case.* Berikut ini merupakan penjelasan dari contoh *use case,* yaitu:

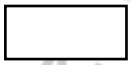
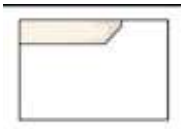
1. Berdasarkan *use case* terdapat lima aktor yaitu *driver, registered owner, maintainer, insurance company,* dan *department of motor vehicles.*

2. Pada *use case* aktor *driver* mendapatkan *case* mengoperasikan kendaraan.
3. Aktor *registered owner* mendapatkan *case* mengasuransikan kendaraan dan mendaftarkan kendaraan ke departermen.
4. Aktor *maintainer* mendapatkan *case* melakukan pemeliharan pada kendaraan.
5. Aktor *insurance company* mendapatkan *case* melakukan pendaftaran asuransi kendaraan dari aktor *registered owner*.
6. Aktor *department of motor vehicles* terdapat *case* menerima pendaftaran yang telah dilakukan oleh *owner*.

2.1.13. *Sequence Diagram*

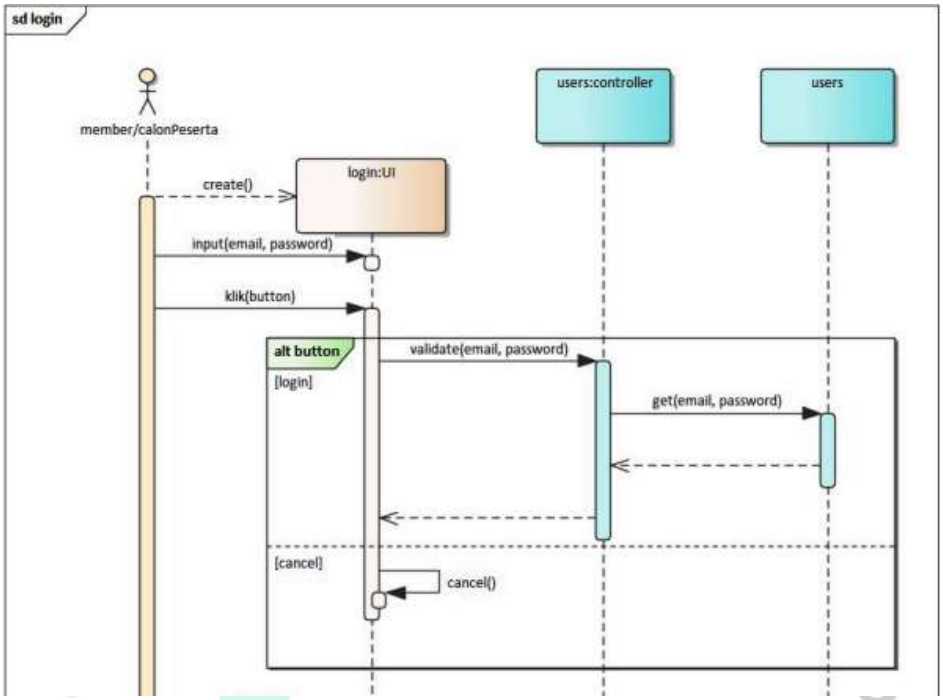
Sequence diagram adalah diagram yang menggambarkan interaksi objek – objek dan pesan – pesan yang ditampilkan secara urutan waktu. Interaksi dispesifikasikan oleh pesan – pesan yang terdapat pada sebuah waktu. Penggambaran *sequence diagram* dapat dilakukan berdasarkan beberapa *level* secara detail dan dengan tujuan yang tergolong berbeda dari langkah – langkah yang dikembangan secara *lifecycle* (Al-Fedaghi, 2021). Berikut ini merupakan komponen yang terdapat pada *sequence diagram* dapat dilihat pada **Tabel 2.3.** di bawah ini.

Tabel 2.3. Komponen *Sequence Diagram*

Simbol	Nama	Keterangan
	*Object Lifeline*	Simbol yang menggambarkan objek yang akan terlibat.
	*Fragment*	Komponen *fragment* dapat digunakan untuk mewakili suatu proses alternatif pada *sequence diagram.* Pada *fragment* terdapat dua proses yang sering digunakan yaitu *loop fragment* dan *alt fragment. Loop fragement* biasanya digunakan untuk kondisi dimana sebuah data diambil secara terus menerut dari suatu objek.

		Sedangkan *alt fragment* digunakan pada proses opsi yang dapat dipilih.
	Aktor	Simbol yang menunjukan adanya pengguna yang akan berinteraksi.
	*Activation Box*	Simbol yang menjelaskan hubungan objek dengan pesan.
	Call Message	Simbol yang menggambarkan garis penghubung untuk mengirim pesan.
	Return Message	Simbol garis alur pengembalian pesan ke objek pemanggil sebelumnya.

Berdasarkan penjelasan definisi dan komponen dari *sequence diagram* di atas. Berikut ini merupakan contoh *sequence diagram* yang dapat dilihat pada **Gambar 2.8.** di bawah ini.



Gambar 2.8. Contoh *Sequence Diagram*

Sumber: (Kemal, 2023)

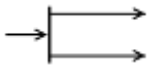
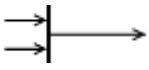
Pada **Gambar 2.8.** di atas merupakan contoh dari *sequence diagram login*. Berikut ini merupakan penjelasan dari contoh *sequence diagram*, yaitu:

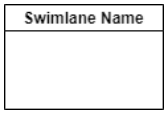
1. Terdapat aktor member / calon peserta yang berinteraksi dengan tampilan halaman *login.*
2. Jika aktor ingin masuk ke dalam sistem maka pengguna harus memasukan email dan *password* pada tampilan halaman *login.*
3. Setelah itu aktor akan menentukan pilihan dengan menekan dua tombol atau *button* yang terdapat pada *fragment* Alt. Tombol tersebut *login* dan *cancel*
4. Jika aktor menekan tombol *login* maka email dan *password* akan divalidasi pada *controller users.* Validasi data tersebut juga akan menarik data email dan *password* pada *class users.*
5. Sedangkan jika tombol *cancel* ditekan maka aktor akan dialihkan kehalaman sebelumnya.

2.1.14. *Activity Diagram*

Diagram aktivitas merupakan gambaran secara visual dari alur suatu sistem yang berisikan aktivitas, tindakan, pilihan, pengulangan, serta *concurrency.* Pada *activity diagram* memiliki beberapa komponen dengan bentuk yang berbeda – beda dan diarahkan berdasarkan urutan aktivitas (Destriana et al., 2021). Berikut ini merupakan komponen dari diagram aktivitas yang dapat dilihat pada **Tabel 2.4.** di bawah ini.

Tabel 2.4. Komponen *Activity Diagram*

Simbol	Nama	Keterangan
	*Initial Node*	Simbol dimana suatu aktivitas akan dimulai.
	*Final Node*	Simbol setelah suatu aktivitas telah selesai dijalankan.
	*Activity*	Simbol yang dapat menggambarkan bagaimana kelas antarmuka saling berinteraksi.
	*Decision*	Simbol yang menunjukan sebuah pilihan dalam suatu tindakan.
	*Merge*	Simbol yang menggabungkan alur setelah dipecah pada *decision.*
	Association	Simbol alur penghubung aktivitas dengan aktivitas lainnya.
	*Synchronization Fork*	Simbol jika satu alur kegiatan berubah menjadi beberapa aliran.
	*Synchronization Join*	Simbol dimana beberapa alur kegiatan berubah menjadi satu alur.

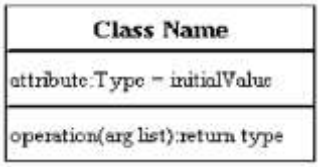
	*Swimlane*	Simbol untuk memisahkan organisasi bisnis yang memiliki tanggung jawab yang berbeda.

2.1.15. *Class Diagram*

Class diagram merupakan penjelasan yang penting dan sudah banyak digunakan dalam sistem dengan konsep OOAD. *Class diagram* menggambarkan struktur statis yang berasal dari beberapa *class* inti pada sistem. Pada diagram kelas menampilkan *method* dan *attribute* di setiap *class* serta menampilkan *relation* atau hubungan di antara *class* – *class* (Paradis et al., 2020). Berikut ini merupakan komponen dari *class diagram*, yaitu (Rahardja & Efana Rahwanto, 2020):

A. *Class*

Komponen *class* merupakan simbol yang menjelaskan himpunan objek yang berisikan *attribute* serta *operation* atau *method.* Berikut ini merupakan contoh penggunaan *class* yang dapat dilihat pada **Gambar 2.9.** di bawah ini.



Gambar 2.9. Contoh *Class*

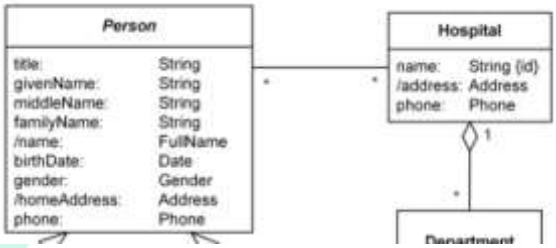
Sumber: (Nasril & Aribah, 2018)

Pada **Gambar 2.9.** di atas merupakan contoh *class*. Dalam penggunaan *class name* dapat diisi sesuai dengan perancangan, sedangkan pada *attribute* diisi dengan definisi *property*, dan pada bagian bawah yaitu *operation* berisikan *methode* – *methode* yang ada pada *class.*

B. *Association*

Gambar 2.10. Komponen Association

Pada **Gambar 2.10.** di atas merupakan komponen *association* adalah simbol garis yang digunakan untuk menghubungkan objek – objek atau *class*. Berikut ini merupakan contoh penggunaan *association* yang dapat dilihat pada **Gambar 2.11.** di bawah ini.



Gambar 2.11. Contoh *Association*

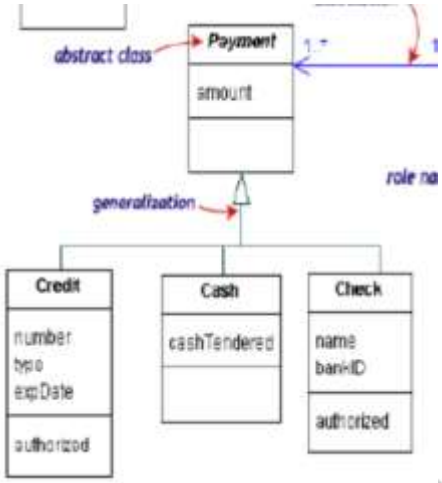
Sumber: (www.sekawanmedia.co.id)

Pada **Gambar 2.11.** di atas merupakan contoh penggunaan *association.* Berdasarkan contoh di atas *class person* berhubungan dengan *class hospital* yang dimana terdapat relasi antara objek *class.*

C. *Generalization*

Gambar 2.12. Komponen Generalization

Pada **Gambar 2.12.** di atas merupakan komponen *generalization.* Komponen tersebut menunjukan hubungan objek anak yang memiliki turunan / mewarisi struktur data dan sifat pada objek induk. Berikut ini merupkan contoh dari *generalization* yang dapat dilihat pada **Gambar 2.13.** di bawah ini.



Gambar 2.13. Contoh *Generalization*

Sumber: (www.ibm.com)

Berdasarkan **Gambar 2.13.** di atas menjelaskan bahwa *class credit*, *cash*, dan *check* merupakan bagian dari *class payment* yang dimana terdapat turunan pada struktur data.

D. *Dependency*

Gambar 2.14. Komponen *Dependency*

Pada **Gambar 2.14.** di atas adalah komponen *dependency*. Komponen ini merupakan sebuah garis hubung relasi pada *class* yang bergantungan dengan *class* lain. Berikut ini merupakan contoh dari penggunaan *dependency* yang dapat dilihat pada **Gambar 2.15.** di bawah ini.

Gambar 2.15. Contoh *Dependency*

Sumber: (blog.visual-paradigm.com)

Berdasarkan **Gambar 2.15.** di atas menjalaskan bahwa sebuah mobil sangat bergantungan dengan bahan bakar atau bensin.

E. *Aggregation*

Gambar 2.16. Komponen *Aggregation*

Pada **Gambar 2.16.** di atas adalah komponen *aggregation.* Komponen ini merupakan hubungan dari semua bagian *class* dan *class* yang terpisah, namun objek dari *class* dapat berdiri sendiri tanpa terpengaruh oleh *class* lain. Berikut ini merupakan contoh dari penggunaan *aggregation* yang dapat dilihat pada **Gambar 2.17.** di bawah ini.

Gambar 2.17. Contoh Aggregation

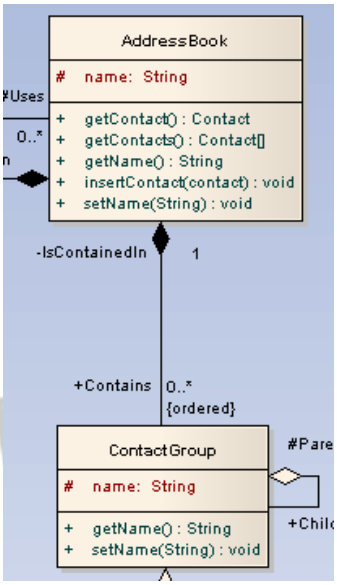
Sumber: (https://sparxsystems.com/)

Berdasarkan **Gambar 2.17.** di atas menjalaskan bahwa *class account* memiliki relasi aggregasi dengan *addressbook*. Jika dijelaskan menjadi satu *account* dapat terdapat banyak atau tidak sama sekali *addressbook,* dan pada *class addressbook* masih dapat berdiri tanpa adanya hubungan dengan *class account.*

F. *Composition*

Gambar 2.18. Komponen *Composition*

Pada **Gambar 2.18.** di atas adalah komponen *composition.* Komponen ini merupakan suatu hubungan dari seluruh *class* dan memiliki sifat kebergantungan kepada *class* lain. Berikut ini merupakan contoh dari penggunaan *composition* yang dapat dilihat pada **Gambar 2.19.** di bawah ini.



Gambar 2.19. Contoh *Composition*

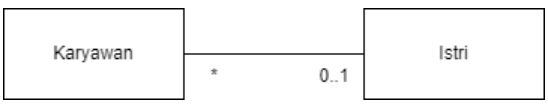
Sumber: (https://sparxsystems.com/)

Berdasarkan **Gambar 2.19.** di atas menjalaskan bahwa pada *class addressbook* memiliki relasi *composition* pada *class contactgrup*. Jika dijelaskan dengan detail setiap *addressbook* terdapat banyak atau tidak sama sekali *contactgrup*, dan pada *class contactgroup* tidak dapat berdiri tanpa adanya *class addressbook.*

Pada *class diagram* akan menunjukan bahwa setiap *class* memiliki hubungan atau relasi antar *class*. Pada setiap relasi tersebut dilengkapi dengan *multiplicity*. *Multiplicity* tersebut lebih dikenal dengan jenis hubungan, *class diagram* memiliki 10 indikator jenis hubungan. Berikut ini merupakan *multiplicity* pada *class diagram*, yaitu (Rahardja & Efana Rahwanto, 2020):

A. Nol sampai satu

Multiplicity nol sampai satu diindikatorkan 0..1 yang biasanya ditempatkan pada garis hubungan antar *class.* Berikut ini merupakan contoh dari *multiplicity* 0..1 yang dapat dilihat pada **Gambar 2.20.** di bawah.



Gambar 2.20. Contoh *Multiplicity 0..1*

Berdasarkan **Gambar 2.20.** di atas menjelaskan bahwa banyak karyawan yang sudah memiliki satu istri dan ada yang belum memiliki istri.

B. Hanya satu

Pada *multiplicity* hanya satu diindikatorkan 1 yang ditempatkan pada garis penghubung antar *class.* Berikut ini merupakan contoh dari *multiplicity* 1 yang dapat dilihat pada **Gambar 2.21.** di bawah.

Gambar 2.21. Contoh *Multiplicity 1*

Berdasarkan **Gambar 2.21.** di atas menjelaskan bahwa satu negara pasti memiliki satu Presiden untuk memerintah negaranya.

C. Nol sampai lebih

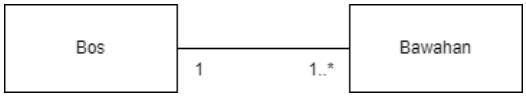
Multiplicity hanya nol sampai lebih diindikatorkan 0..* yang biasanya terdapat pada garis relasi antar *class.* Berikut ini merupakan contoh dari *multiplicity* 0..* yang dapat dilihat pada **Gambar 2.22.** di bawah.

Gambar 2.22. Contoh *Multiplicity 0..**

Berdasarkan **Gambar 2.22.** di atas menjelaskan bahwa satu karyawan sudah memiliki beberapa anak dan ada juga yang belum memiliki anak.

D. Satu sampai lebih

Multiplicity satu sampai lebih diindikatorkan 1..* yang biasanya ditempatkan pada garis hubungan antar *class.* Berikut ini merupakan contoh dari *multiplicity* 1..* yang dapat dilihat pada **Gambar 2.23.** di bawah.



Gambar 2.23. Contoh *Multiplicity 1..**

Berdasarkan **Gambar 2.23.** di atas menjelaskan bahwa disuatu organisasi terdapat satu bos yang memiliki satu sampai lebih anak buah.

E. Banyak

Multiplicity banyak diindikatorkan sebagai bintang (*) yang biasanya ditempatkan pada garis relasi antar *class* satu dengan *class* yang lainnya. Berikut ini merupakan contoh dari *multiplicity* * yang dapat dilihat pada **Gambar 2.24.** di bawah.

Gambar 2.24. Contoh *Multiplicity* Banyak

Berdasarkan **Gambar 2.24.** di atas menjelaskan bahwa disuatu perusahan memiliki banyak karyawan.

F. Lebih dari satu

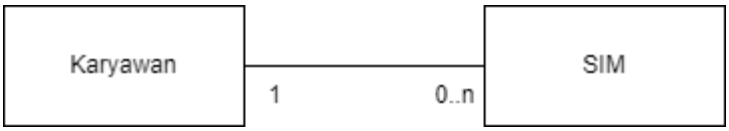
Pada *multiplicity* lebih dari satu diindikatorkan (n) yang biasanya ditempatkan pada garis penghubung antar *class* satu dengan *class* yang lainnya. Berikut ini merupakan contoh dari *multiplicity* n yang dapat dilihat pada **Gambar 2.25.** di bawah.

Gambar 2.25. Contoh *Multiplicity* n

Pada **Gambar 2.25.** di atas menjelaskan bahwa pada suatu perusahaan terdapat lebih dari satu divisi.

G. Lebih dari sama dengan satu

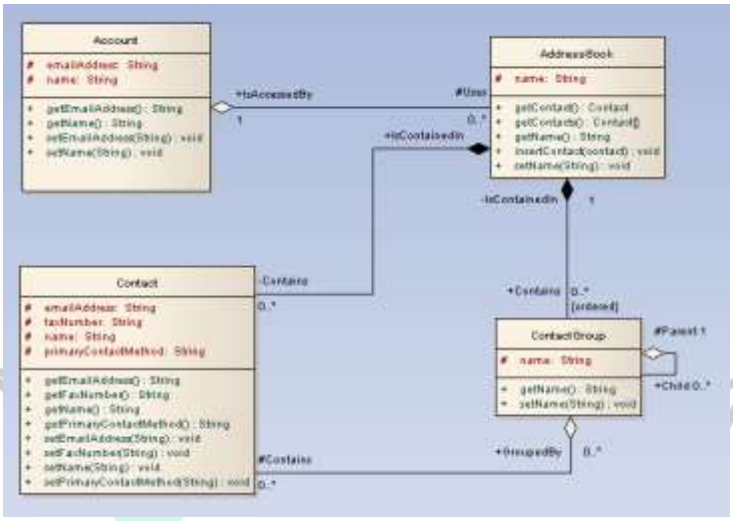
Pada *multiplicity* lebih dari sama dengan satu diindikatorkan menjadi 0..n yang biasanya diletakan pada garis relasi antar *class* satu dengan *class* yang lainnya. Berikut ini merupkan contoh dari *multiplicity* 0..n yang dapat dilihat pada **Gambar 2.26.** di bawah ini.



Gambar 2.26. Contoh Multiplicity 0..n

Pada **Gambar 2.26.** di atas menjelaskan bahwa satu karyawan yang tidak memiliki SIM dan memiliki lebih dari satu SIM.

Berdasarkan penjelasan konsep dasar dari *class diagram* di atas. Berikut ini merupakan contoh *class diagram* yang dapat dilihat pada **Gambar 2.27.** di bawah ini.



Gambar 2.27. Contoh Class Diagram

Sumber: (www.sparxsystems.com)

Pada **Gambar 2.27.** di atas adalah contoh *class diagram* data pribadi. Berikut ini merupakan penjelasan dari contoh *class diagram*, yaitu:

1. Terdapat *class account* yang memiliki atribut *emailAddress* dengan tipe data *string* dan atribut *name* dengan tipe data *string.* Serta memiliki beberapa *method*, yaitu *getEmailAddress, getName, setEmailAddress,* dan *setName.*
2. Terdapat *class addressBook* yang memiliki atribut *name* dengan tipe data *string*. Serta terdapat beberapa *method*, yaitu *getContact, getContacts, getName, insertContact,* dan *setName.*
3. Terdapat *class contact* yang memiliki atribut *emailAddress* dengan tipe data *string*, *faxNumber* dengan tipe data *string*, *name* dengan tipe data *string*, dan *primaryContactMethod* dengan tipe data *string*. Serta memiliki beberapa *method*, yaitu *getEmailAddress, getFaxNumber, getName, getPrimaryContactMethod, setEmailAddress, setFaxNumber, setName*, dan *setPrimaryContactMethod.*
4. Terdapat *class contactGroup* yang memiliki atribut *name* dengan tipe data *string*. Serta memiliki *method getName* dan *setName.*

5. Pada *class addressBook* memiliki relasi *aggregation* pada *class account* yang dimana *account* sebagi *parent* sedangkan *addressBook* sebagai *child.* Pada relasi tersebut memiliki *multiplicity* pada masing – masing *class. Class account* terdapat *multiplicity* hanya satu dan *addressBook* terdapat nol sampai lebih. Hal tersebut berarti dari satu *account* terdapat nol sampai lebih *addressBook*.
6. Pada *class contactGroup* memiliki relasi *composition* pada *class addressBook* yang dimana *addressBook* sebagai *parent* sedangkan *contactGroup* sebagai *child.* Pada relasi tersebut memiliki *multiplicity* pada masing – masing *class. Class contactGroup* terdapat *multiplicity* nol sampai lebih dan *addressBook* terdapat *multiplicity* hanya satu. Jika dibaca maka satu *addressBook* terdapat nol sampai lebih *contactGroup* dan *class contactGroup* tidak dapat terbentuk tanpa adanya *class addressBook.*
7. Pada *class contact* memiliki relasi dengan dua *class* yaitu *addressBook* berhubungan *composition* dan *contactGroup* berhubungan *aggregation.* Berdasarkan contoh *class contact* sebagai *child* sedangkan *class addressBook* dan *contactGroup* sebagai *parent.* Pada relasi tersebut memiliki *multiplicity* pada masing – masing *class.* Pada kedua relasi *class contact* memiliki *multiplicity* nol sampai lebih. Sedangkan *class addressBook* dan *contactGroup* memiliki *multiplicity* nol sampai lebih. Hal tersebut menjelasakan bahwa terdapat nol sampai lebih *contact* yang berisikan nol sampai lebih *addressBook* dan *contact* tidak dapat berdiri sendiri tanpa adanya *addressBook.* Sedangkan pada relasi *contact* dengan *contactGroup* menjelaskan bahwa terdapat nol sampai lebih *contact* yang berisikan nol sampai lebih *contactGroup* dan *class contact* masih dapat terbentuk tanpa adanya *contactGroup.*

2.1.16. *Black Box Testing*

Pengujian pada sebuah *software* merupakan suatu hal yang sangat penting, dengan dilakukannya pengujian maka akan diketahui adanya kekurangan, *bug* ataupun *error* pada sistem. *Black box testing* merupakan metode pengujian sistem atau *software* yang dilakukan tanpa mementingkan

detail dari sistem. Pengujian *black box* dilakukan dengan memeriksa hasil *output* berdasarkan nilai yang telah di-*input*, tanpa harus mengetahui *source code* sistem dalam mengeluarkan hasil *output* (Ningrum et al., 2019).

2.2. Tinjauan Studi

Pada penulisan ini mengambil beberapa referensi dari jurnal yang berhubungan dengan penelitian, berikut ini merupakan *literature review* dari lima jurnal yang dapat dilihat pada **Tabel 2.5.** di bawah ini.

Tabel 2.5. Hasil Studi Literatur

Judul	Penulis	Ringkasan	Simpulan
ANALISA DAN PERANCANGAN SISTEM TATA KELOLA PARKIR CERDAS DI KOTA PINTAR JAKARTA	Arman Syah Putra, 2020	Pada jurnal dengan judul tersebut membahas permasalah pada pemesanan tempat parkir yang tidak ada sistemnya. Sistem yang ada pada saat ini yaitu dengan mencari tempat parkir secara *on the spot* atau langsung ke lokasi parkir. Metode yang digunakan pada penelitian ini yaitu studi literature dan studi keperpustakaan. Penelitian ini mengangkat tentang tata kelola parkir cerdas yang belum ada sampai sekarang, sistem yang ada hanya mengetahui beberapa tempat parkir yang kosong. Namun dengan sistem yang diusulkan ini maka	Jurnal dengan judul Analisa dan Perancangan Sistem Tata Kelola Parkir Cerdas di Kota Pintar Jakarta memiliki kecocokan pada penelitian ini karena berubungan dengan sistem *booking* atau reservasi. Namun pada jurnal yang ditulis oleh Arman Syah Putra tahun 2020 penulis menjelaskan bahwa sistem akan dirancang berbasis android. Hal tersebut berbeda dengan penelitian ini yang merancang sistem reservasi berbasis web.

		dapat mempermudah pemesanan tempat parkir jika ingin pergi kesuatu tempat. Dari semua hal yang sudah dilakukan maka penulis menganalisa penelitan dengan menuliskan beberapa kelebihan dan kekurangan dari sistem parkir cerdas, berikut ini merupakan ke dua hal tersebut: - Kelebihan sistem parkir cerdas • Pemesanan tempat parkir akan lebih mudah. • Mengurangi waktu pencarian tempat parkir. • Mengurangi penggunaan uang tunai. • Penggunaan sistem mengurangi pekerja penjaga gardu parkir.	

		- Kekurangan sistem parkir cerdas • Harus mempunyai *smartphone*. • Penggunaan uang digital di setiap transaksi • Biaya untuk membeli dan pemasangan alat dan sistem mahal. Dari hasil analisa dapat disimpulkan bahwa sistem yang diusulkan dengan model pemesanan tempat parkir dari jauh, dengan metode sistem aplikasi android maka dapat mempermudah para pengendara untuk mencari tempat parkir dengan memesan tempat parkir, dan diharapkan dengan sistem ini dapat membantu masyarakat dan membuat lebih tertib dalam memarkirkan	

		kendaraannya dengan teratur.	
System Information Application Meeting Room PT Pertamina Geothermal Energy Area Kamojang	Anisa Melinda Rahmayunita, 2019	Jurnal dengan judul *System Information Application Meeting Room* PT Pertamina Geothermal Energy Area Kamojang membahas tentang permasahan tentang proses peminjaman ruang rapat di PT Pertamina Geothermal Energy di wilayah Kamojang. Pada jurnal tertulis masalah yang terjadi ialah PT Pertamina Geothermal Energy wilayah Kamojang tidak memiliki informasi data yang akurat dan terpercaya mengenai jadwal ruang rapat yang digunakan, hal tersebut dikarenakan tidak adanya pencatatan atau dokumentasi pada proses peminjaman ruang rapat maka sering terjadi kekeliruan di dalamnya. Berdasarkan permasalahan yang tertulis pada jurnal	Pada jurnal tersebut membahas tentang perancangan *System Information Application Meeting Room* PT Pertamina Geothermal Energy Area Kamojang atau SIAMIR. Pada jurnal tersebut memiliki kecocokan dengan penelitian ini yang membahas tentang perancangan sebuah sistem reservasi. Pada jurnal juga menjelaskan bahwa perancangan dilakukan dengan menggunakan model *Prototype, Unified Model Language* (UML), serta menggunakan bahasa pemograman PHP dengan menggunakan *framework* CodeIgniter. Sedangkan database yang digunakan untuk perancangan pada jurnal menggunakan MySql. Beberapa hal

		maka perlu adanya sistem yaitu *System Information Meeting Room* atau SIAMIR yang dapat membantu karyawan untuk melakukan pengecekan ketersediaan ruang rapat, melakukan peminjaman ruang rapat, dan pemesanan pada fasilitas yang telah disediakan. Perancangan sistem yang dibangun menggunakan model *Prototype*, *Unified Model Language* (UML), dan menggunakan bahasa pemograman PHP dengan menggunakan *framework* CodeIgniter serta database menggunakan MySql. Pada tahap pengujian sistem peneliti menggunakan metode *Black Box Testing* dan *User Acceptance Test* (UAT).	tersebut juga akan tercantum dalam penelitian ini.
Sistem Informasi Pemesanan Ruang *Meeting* Berbasi Web App Menggunakan *Framework*	Khasan Asrori dan Ely Nuryani, 2021	Pada jurnal yang disusun oleh Khasan Asrori dan Ely Nuryani membahas permasalahan kurang	Jurnal dengan judul Sistem Informasi Pemesanan Ruang Meeting Berbasi Web App Menggunakan

Codeigniter Pada PT. Barata Indonesia – Cilegon, Banten		efektinya melakukan pemesanan secara manual dengan menghubungi petugas ruangan rapat untuk menanyakan ketersediaan ruang rapat melalui telepon. Hal ini dikarenakan pemesan tidak dapat mengetahui ruangan mana saja yang tersedia untuk dapat digunakan dan kapasitas orang yang dapat ditampung di dalam ruangan. Maka dari itu solusi untuk pemasalah yang terjadi di PT. Batara Indonesia di wilayah Cilegon, Banten dirancang sebuah aplikasi untuk memudahkan pemesanan ruang rapat. Pada jurnal tersebut menggunakan metode penelitian observasi dan wawancara yang bertujuan untuk mendapatkan informasi yang akurat. Sedangkan untuk pengembangan sistem penulis menggunakan metode *System Development Life*	Framework Codeigniter Pada PT. Barata Indonesia – Cilegon, Banten memiliki kecocokan dengan penelitian ini. Kecocokan tersebut terletak pada perancangan sistem informasi reservasi dan menggunakan pengembangan sistem dengan metode *System Development Life Cycle* (SDLC) *waterfall.*

		Cycle (SDLC) atau sering disebut dengan *waterfall.*	
MRAPAT Untuk Sistem Manajemen Ruang Rapat, Absensi, dan Notulen di PT. PLN Unit Wilayah Sumbar	Afdal dan Hidra Amnur, 2020	Pada jurnal membahas tentang pengecekan informasi ketersediaan ruang rapat pada perusahaan masih dilakukan dengan cara menghubungi pihak pengelola melalui telepon, proses absensi rapat juga masih dilakukan manual menggunakan kertas, dan penulisan pembahasan serta pembagian hasil rapat masih dilakukan oleh karyawan. Berdasarkan proses tersebut dinilai sangat tidak efektif, karena untuk melakukan pengecekan dan konfirmasi membutuhkan waktu yang lama, ketika ada yang ingin melakukan perpindahan ruang rapat akan menyebabkan bentrok dengan jadwal lain dan agenda menjadi tertunda, lalu karyawan tidak dapat mengetahui ketersediaan ruang	Jurnal yang disusun oleh Afdal dan Hidra Amnur pada tahun 2020 memiliki kecocokan dengan penelitian ini terkait dengan perancangan sistem informasi reservasi ruang rapat. Namun pada jurnal tersebut memiliki *scope* yang luas. *Scope* tidak memiliki keterkaitan dengan penelitian yang terdapat pada jurnal yaitu absensi dan notulen.

		rapat yang dapat digunakan, sedangkan pelaksanaan absensi rapat terlalu memakan banyak waktu untuk melakukan penandatanganan oleh karyawan dan data dari absensi sudah didapatkan perlu *input* kembali oleh pihak sekretaris ke komputer, hal tersebut membutuhkan pengerjaan dua kali dan berpeluang adanya kecurangan pada absensi, pada notulen juga perlu pengerjaan dua kali dan hasil rapat perlu di salin ke dalam *Microsoft Word* untuk dikirim ke email kepada peserta rapat. Maka dari itu pada jurnal ditentukan untuk merancang sebuah sistem dengan nama MRapat berbasis website dan aplikasi andorid. Sistem tersebut dibuat untuk membantu dalam mengelola ruangan, absensi, dan notulen agar lebih efisien. Adanya sistem ini dapat memanajemen	

		informasi tentang ketersediaan ruangan, absensi rapat, dan notulen dengan baik. Pada jurnal tertulis metode untuk pembuatan sistem yaitu pengumpulan data, analisis, dan perancangan dan untuk mengimplementasikan sistem dengan berbasis website penulis memanfaatkan *framework* Laravel dan bahasa pemograman PHP (*Hypertext Preprocessor*) sedangkan sistem berbasis aplikasi android menggunakan Android Studio dengan bahasa pemogromanan JAVA.	
Rancang Bangun Aplikasi *Booking Meeting Room* Berbasis Android Dengan Pencarian Nama Ruangan Menggunakan Algortima Brute Force (Studi Kasus: Universitas Mercu Buana	Muhamad Ridwan dan Eliyani, 2019	Jurnal dengan judul "Rancang Bangun Aplikasi Booking Meeting Room Berbasis Android Dengan Pencarian Nama Ruangan Menggunakan Algoritma Brute Force (Studi Kasus: Universitas Mercu Buana)" disusun oleh Muhamad Ridwan dan	

		Eliyani pada tahun 2019, membahas permasahan pemesanan ruang rapat yang dilakukan dengan mengunjungi lokasi dan menanyakan ketersediaan ruangan melalui telepon di nilai tidak efektif dan apabila karyawan ingin memesan ruangan dalam waktu dekat dan sesuai dengan kapasitas yang diinginkan. Maka solusi dari masalah yang terjadi dengan membuat sistem pemesanan ruang rapat berbasis android dengan menggunakan algoritma Brute Force. Dalam teknik pengumpulan data peneliti menggunakan metode observasi, wawancara, dan studi Pustaka. Sedangkan untuk model pengembangan sistem peneliti menggunakan Model *Waterfall*, pada model tersebut terbagi menjadi beberapa tahap, yaitu :	

		- Analisa Kebutuhan *Software* - *Design* Spesifikasi - *Code Generation* - *Support*	