BAB 2
TINJAUAN PUSTAKA

2.1 Profil Perusahaan

EGS hearT Group adalah perusahaan berbentuk Perseroan Terbatas dengan bisnis utama sebagai penyedia *facility service* yang berkualitas dan terpercaya. Kami merupakan sekelompok profesional yang sudah berpengalaman di bidang keamanan dan industri kebersihan.

Didirikan sejak 2011, kami telah melayani lebih dari seratus klien dengan cakupan kerja ribuan meter persegi setiap harinya. Klien kami sangatlah beragam, terdiri dari bisnis menengah dengan layanan mingguan hingga skala besar yang mempekerjakan ratusan staf. Hal ini menujukkan kemampuan kami untuk menciptakan solusi layanan kebersihan dan keamanan yang menjawab kebutuhan pada beragam tingkatan.

Logo PT. EGS hearT group dapat di lihat pada gambar 2.1.

Gambar 2.1 Logo PT. EGS HearT Group

2.2 Landasan Teori

2.2.1 *Task management*

Task management adalah aktivitas di mana seorang *task assigner* (*Manager*) melacak sebuah *task* sepanjang siklus hidupnya dan membuat keputusan berdasarkan *progress* atau kemajuan dari *task* tersebut. *Task life cycle* atau siklus hidup *task* terdiri dari sembilan state atau keadaan yaitu *Ready* , *Assigned*, *Terminated*, *Expired*, *Forwarded*, *Failed*, *Finished*. Sebuah tugas juga dibedakan oleh kompleksitas, yaitu dari rendah ke tinggi. Dalam hal ini *task assigner* bertanggung jawab untuk membuat, menetapkan, memprior*itask*an dan memantau tugas untuk memastikan *task* atau tugas selesai pada waktunya. Maka dari itu salah satu tujuan *task management* adalah dapat membantu pemimpin dalam mengelola *task* atau tugas serta memonitor setiap *task*. Dasar untuk mengatur dan mengelola tugas yaitu dengan menggunakan fungsi seperti pembuatan tugas, perencanaan, penugasan, pelacakan dan pelaporan [12].

Pada *task management* digunakan rumus untuk mencari efisiensi pengerjaan tugas, berikut rumus untuk menghitung efisiensi pengerjaan tugas :

$$\text{Efisiensi} = \frac{\sum Ti}{(n-1)\ .\ Ts}$$

Dimana :

Ti = Waktu per Tugas

n = Jumlah Tugas

Ts = Waktu Siklus (waktu yang tersedia perhari / tugas yang harus diselesaikan perhari)

2.2.1.1 *Task*

Task atau tugas adalah sesuatu pekerjaan yang harus dikerjakan atau dieselesaikan. Akan tetapi *task* atau tugas yang dimaksud dalam tugas akhir ini adalah bagian terkecil dari pekerjaan yang dapat dikenali dan berfungsi sebagai unit kerja [13].

2.2.1.2 Manajemen

Definisi Manajemen adalah suatu seni mengarahkan orang lain untuk mencapai tujuan utama dalam suatu organisasi melalui proses perencanaan (*Planning*), pengorganisasian (*Organizing*), dan mengelola (*Controlling*) sumber daya manusia dengan cara efektif dan efisien [13].

Pengertian dan Definisi Manajemen menurut Para Ahli sebagai berikut :

1) Manajemen adalah adalah suatu proses yang berbeda terdiri dari *planning*, *organizing*, *actuating*, dan *controlling* yang dilakukan untuk mencapai tujuan yang ditentukan dengan menggunakan manusia dan sumber daya lainnya (George R. Terry, 1997).
2) Manajemen adalah suatu seni yang produktif yang didasarkan pada suatu pemahaman ilmu, ilmu dan seni tidaklah bertentangan, namun masing masing saling melengkapi (Koontz).
3) Ilmu Manajemen merupakan proses dalam membuat suatu perencanaan, pengorganisisasian, pengendalian serta memimpin berbagai usaha dari anggota entitas atau organisasi dan juga mempergunakan semua sumber daya yang dimiliki untuk mencapai tujuan yang ditetapkan (Stoner).
4) Manajemen sebagai sebuah rangkaian tindakan tindakan yang dilakukan oleh para anggota organisasi dalam upaya mencapai sasaran organisasi. proses merupakan suatu rangkaian aktivitas yang dijalankan dengan sistematis (Wilson).
5) Manajemen adalah sebuah seni dalam mencapai tujuan yang diinginkan yang dilaksanakan dengan usaha orang yang lain (Lawrance A Appley)
6) Manajemen sebagai suatu seni, tiap tiap pekerjaan bisa diselesaikan dengan orang lain (Mary Parker F).

2.2.2 Gamifikasi

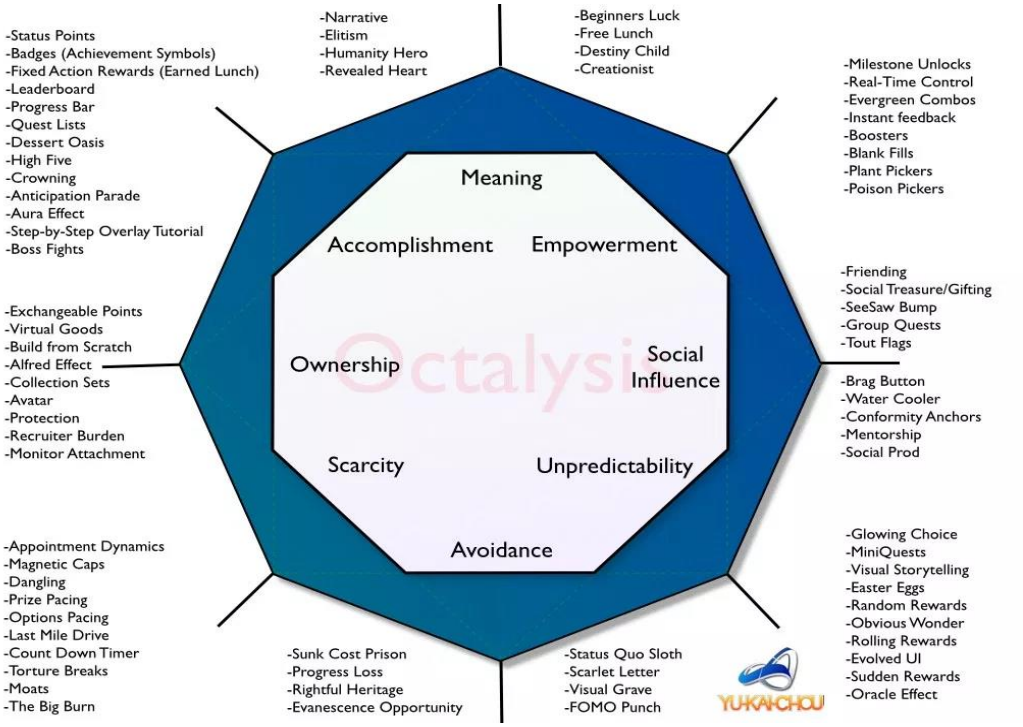
Gabe Zichermann pada buku *Gamification By Design*, mengatakan bahwa gamifikasi merupakan proses dari cara berpikir seperti *game* dan memasukan komponen komponen yang ada pada *game* untuk melibatkan pengguna dan menyelesaikan masalah tertentu. Gamifikasi terbagi menjadi 2, yaitu gamifikasi *by*

content dan gamifikasi struktural. Gamifikasi *by content* merupakan gamifikasi dengan ciri memasukan sebuah *story* atau cerita didalamnya, cocoknya gamifikasi ini dipakai untuk media pembelajaran. Sedangkan gamifikasi struktural hanya mengambil komponen komponen di dalam *game*, seperti poin, *leaderboard*, *reward*, *level*, *badge* dan lain lain. Gamifikasi struktural biasa digunakan untuk aplikasi yang menyangkut alur sebuah pekerjaan [14]. Komponen-komponen yang ada pada *game* disebut juga *game mechanics*.

Gambar 2.2 Konsep Gamifikasi

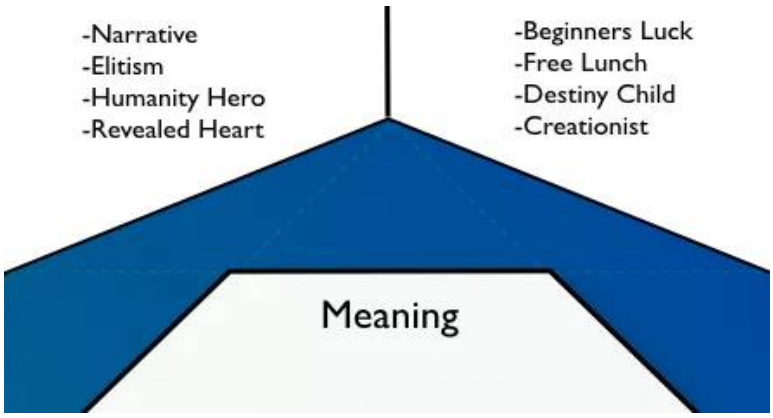
2.2.3 *Octalysis Framework*

Octalysis merupakan metode gamifikasi yang dikembangkan oleh Yu-kai Chou pada tahun 2015. Metode Octalysis memiliki 2 level, dimana level pertama merupakan analisis elemen game dari Octalysis Framework, sedangakan level kedua penerapan elemen game pada 4 fase yang sudah disediakan. Octalysis diambil dari kerangka gamifikasi yang dirancang menggunakan 8 Core Drive. Berikut ini kerangka gamifikasi dengan metode octalysis pada level pertama [15].



Gambar 2.3 Kerangka *Octalysis*

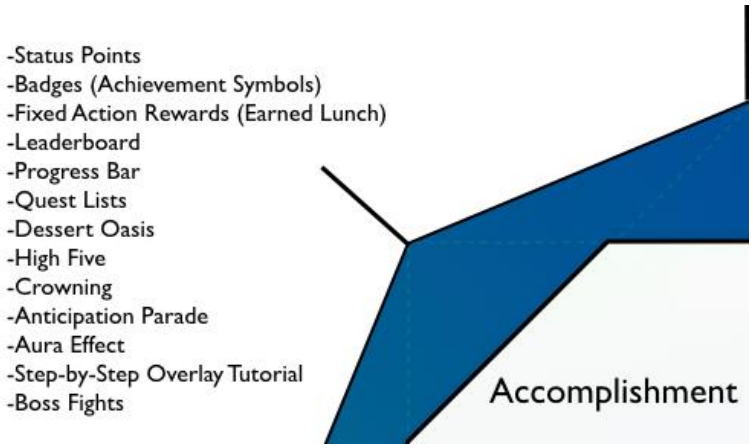
2.2.3.1 *Epic Meaning and Calling*



Gambar 2.4 *Epic Meaning and Calling*

Epic Meaning and Calling adalah drive yang digunakan untuk mengukur dan memahami seberapa besar tingkat kepercayaan seseorang dalam melakukan pekerjaan tersebut apakah yang dilakukannya berdasarkan keinginannya sendiri atau terpilih untuk melakukan pekerjaan itu.

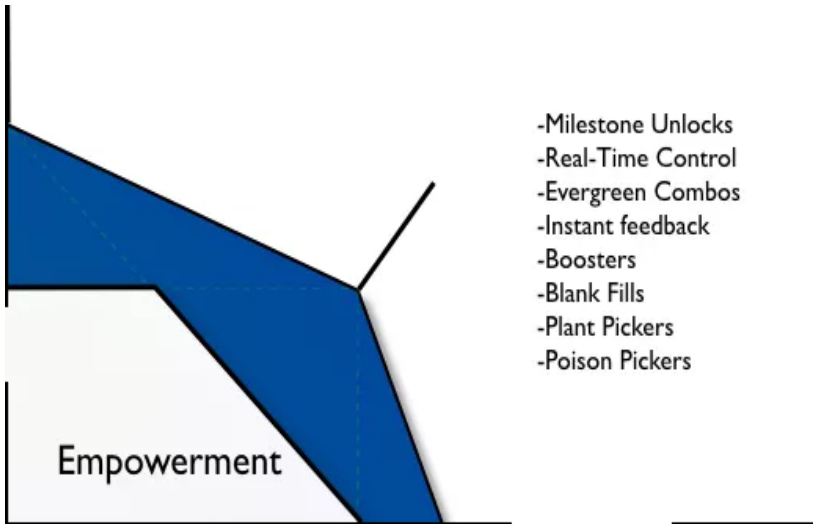
2.2.3.2 *Development and Accomplishment*



Gambar 2.5 ***Develpoment & Accomplishment***

Development and Accomplishment adalah dorongan internal untuk membuat kemajuan, mengembangkan keterampilan, dan akhirnya mengatasi tantangan. Kata "tantangan" di sini sangat penting, karena lencana atau piala tanpa tantangan sama sekali tidak berarti.

2.2.3.3 *Empowerment of Creavity and Feedback*

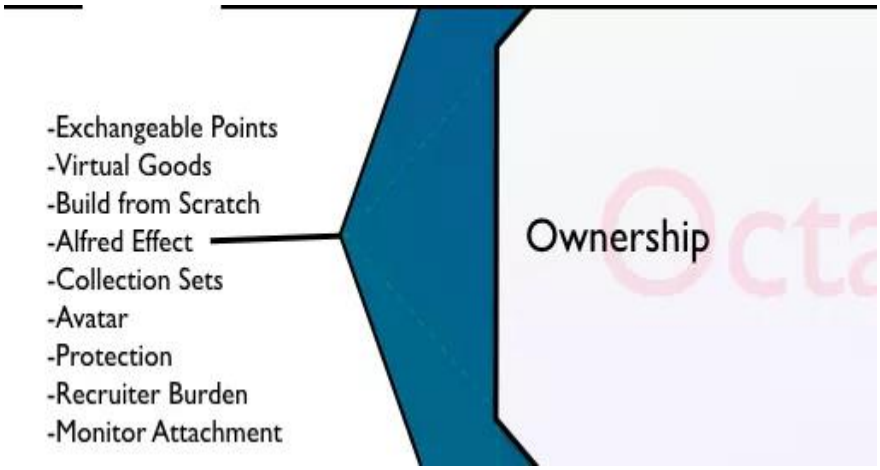


Gambar 2.6 ***Empowerment of Creavity and Feedback***

Empowerment of Creavity and Feedback adalah drive yang melibatkan pengguna untuk menggunakan kreativitasnya untuk menyelesaikan masalah baik dengan cara mencari cara baru atau mencoba kombinasi yang berbeda. Pengguna

tidak hanya membutuhkan cara untuk mengekspresikan kreativitas mereka, tetapi juga perlu melihat hasil kreatifitas mereka dengan menerima umpan balik. Inilah sebabnya mengapa orang bermain Lego dan Minecraft lebih menyenangkan karena dari kreatifitas tersebut orang juga mendapatkan umpan balik baik berupa pujian.

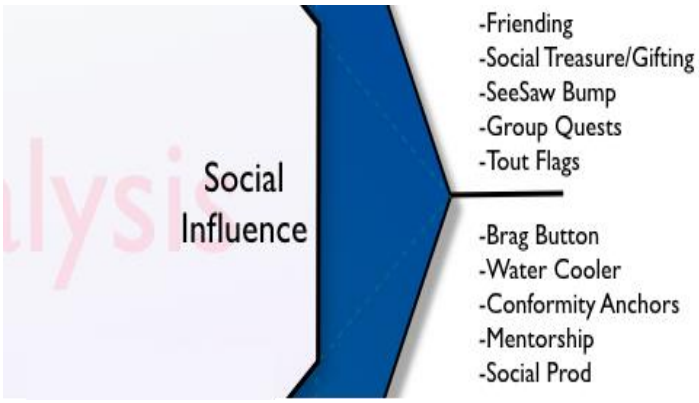
2.2.3.4 *Ownership and Possesion*



Gambar 2.7 ***Ownership and Possesion***

Ownership and Possesion adalah drive yang mengacu pada pengguna untuk merasa bahwa mereka memiliki atau dapat mengendalikan sesuatu. Ketika seseorang merasa memiliki sesuatu, mereka dengan sendirinya ingin meningkatkan dan memperbaiki apa yang mereka miliki. Sebagai contoh seseorang yang memiliki kekayaan dalam bentuk uang akan menggunakan kekayaan tersebut untuk meningkatkan jumlah uangnya dan mengendalikan uang tersebut untuk digunakan sesuai dengan kebutuhan dan keinginannya.

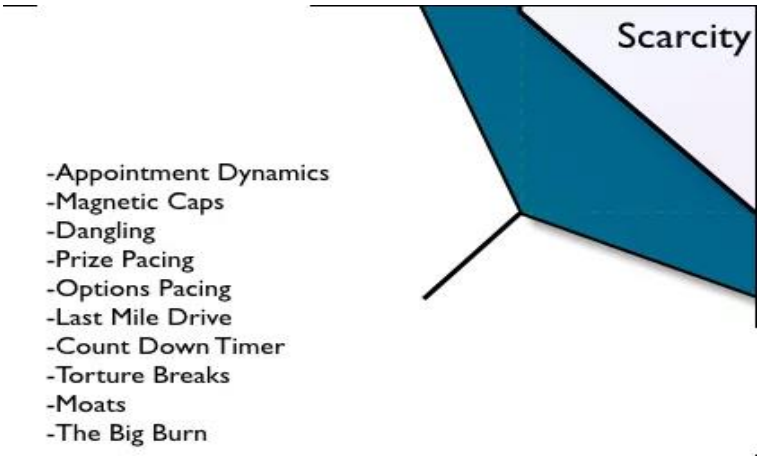
2.2.3.5 *Social Influence and Relatedness*



Gambar 2.8 ***Social Influence and Relatedness***

Social Influence and Relatedness adalah drive yang menggabungkan semua elemen sosial, termasuk: bimbingan, penerimaan, tanggapan sosial, persahabatan, serta persaingan dan kecemburuan. Ketika Anda melihat seorang teman yang luar biasa pada suatu keterampilan atau memiliki sesuatu yang luar biasa, Anda menjadi terdorong untuk mencapai tingkat yang sama. Juga, ini termasuk dorongan yang harus kita dekatkan dengan orang, tempat, atau peristiwa yang dapat kita hubungkan. Jika Anda melihat produk yang mengingatkan Anda akan masa kecil Anda, rasa nostalgia kemungkinan akan meningkatkan peluang Anda untuk membeli produk tersebut. Core drive ini juga relatif banyak dipelajari, karena banyak perusahaan saat ini menempatkan banyak prioritas untuk mengoptimalkan strategi sosial online mereka.

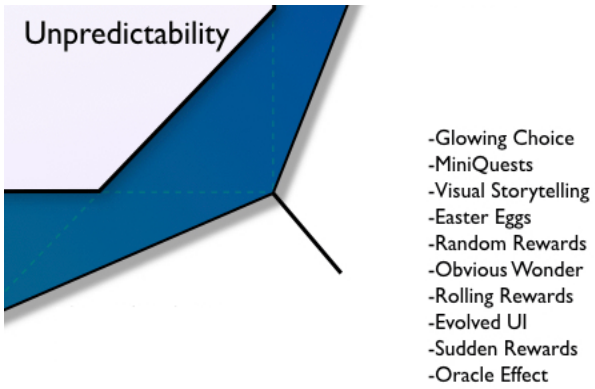
2.2.3.6 *Scarcity and Imaptience*



Gambar 2.9 *Scarcity and Imaptience*

Scarcity and Imaptience adalah drive yang melibatkan pemain untuk menginginkan sesuatu karena tidak dapat memilikinya. Banyak game memiliki Janji Temu Dinamika (kembali 2 jam kemudian untuk mendapatkan hadiah Anda) - fakta bahwa orang tidak bisa mendapatkan sesuatu sekarang memotivasi mereka untuk memikirkannya sepanjang hari. Ini adalah Core Drive yang digunakan oleh Facebook ketika pertama kali dimulai: pada awalnya itu hanya untuk Harvard. Kemudian dibuka untuk beberapa sekolah bergengsi lainnya, dan akhirnya semua perguruan tinggi. Ketika akhirnya terbuka untuk semua orang, banyak orang ingin bergabung karena mereka sebelumnya tidak bisa masuk.

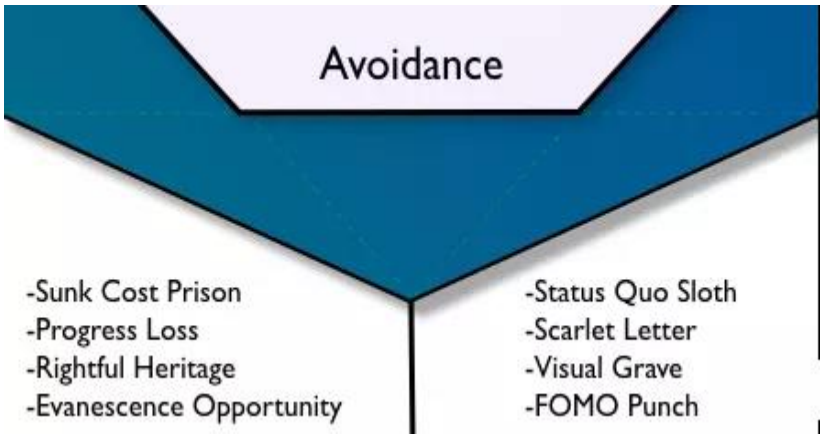
2.2.3.7 *Unpredictability & Curiosity*



Gambar 2.10 *Unpredictability & Curiosity*

Secara umum, Unpredictability & Curiosity adalah dorongan pemain untuk mencari tahu apa yang akan terjadi selanjutnya. Jika Anda tidak tahu apa yang akan terjadi, otak Anda bergerak dan Anda sering memikirkannya. Banyak orang menonton film atau membaca novel karena dorongan ini. Namun, dorongan ini juga merupakan faktor utama di balik kecanduan judi. Juga, drive inti ini digunakan setiap kali perusahaan menjalankan program undian atau lotre untuk melibatkan pengguna. Eksperimen Box Skinner yang sangat kontroversial, di mana seekor hewan secara irasional sering menekan tuas karena hasil yang tidak dapat diprediksi, secara eksklusif merujuk pada penggerak inti dari Unpredictability & Curiosity, meskipun banyak yang salah mengartikannya sebagai penggerak di balik poin, lencana, dan mekanisme papan peringkat pada umumnya.

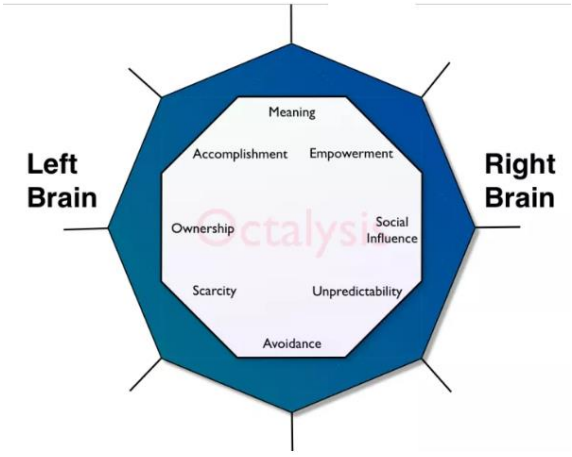
2.2.3.8 *Loss and Avoidance*



Gambar 2.11 ***Loss and Avoidance***

Drive yang terakhir ini mengacu pada motivasi pemain untuk menghindari sesuatu yang negatif. Contohnya adalah menghindari rintangan jurang agar tidak jatuh dan game over. Motivasi ini mengacu pada ketakutan dan kewaspadaan pengguna agar dapat melanjutkan permainan.

2.2.4 *Left Brain vs Right Brain Drives*

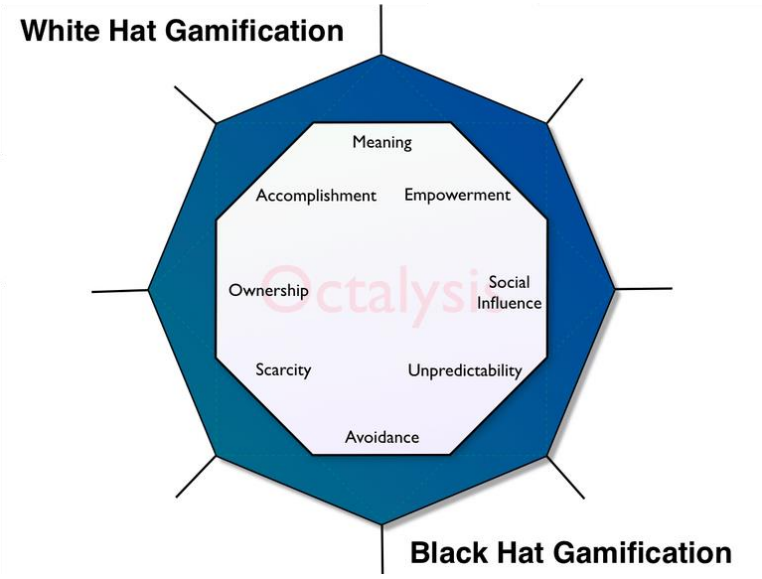


Gambar 2.12 ***Left Brain vs Right Brain Drives***

Dalam Octalysis, Core drives di sebelah kanan adalah right brain core drives, yang lebih terkait dengan kreativitas, ekspresi diri, dan aspek sosial. core drives di sebelah kiri adalah left brain core drives, yang lebih terkait dengan logika, perhitungan, dan kepemilikan.

Menariknya, Drive core otak kiri adalah Motivator Ekstrinsik : Anda termotivasi karena Anda ingin mendapatkan sesuatu, apakah itu tujuan, barang, atau apa pun yang tidak dapat Anda peroleh; di sisi lain, Drive inti otak kanan adalah Motivator Intrinsik : Anda tidak memerlukan tujuan atau hadiah untuk menggunakan kreativitas Anda, bergaul dengan teman-teman, atau merasakan ketegangan ketidakpastian yang tidak dapat diprediksi - kegiatan itu sendiri memberi penghargaan dengan sendirinya [15].

2.2.5 *White Hat vs Black Hat Gamification*



Gambar 2.13 ***White Hat vs Black Hat Gamification***

Elemen lain yang perlu diperhatikan dalam Octalysis adalah Core Drives teratas dalam octagon dianggap sebagai motivator yang sangat positif, sedangkan Core Drives bagian bawah dianggap motivator negatif.

Teknik yang memanfaatkan Core Drive bagian atas disebut "White Hat Gamification" , sedangkan teknik yang memanfaatkan Core Drive bagian bawah disebut "Black Hat Gamification".

Jika sesuatu menarik karena memungkinkan mengekspresikan kreativitas Anda, membuat Anda merasa sukses melalui penguasaan keterampilan, dan memberi Anda makna makna yang lebih tinggi, itu membuat pengguna merasa sangat baik dan kuat.

Di sisi lain, jika Anda selalu melakukan sesuatu karena Anda tidak tahu apa yang akan terjadi selanjutnya, Anda terus-menerus takut kehilangan sesuatu, atau karena ada hal-hal yang tidak dapat Anda miliki, walaupun Anda akan tetap sangat termotivasi. untuk mengambil tindakan, seringkali dapat meninggalkan rasa tidak enak di mulut Anda [15].

2.2.6 *Octalysis Framework Level II*

Octalysis Level II dibagi menjadi 4 fase yaitu discovery, onboarding, scaffolding dan endgame yang dapat dilihat pada daftar berikut ini:

1. Fase *Discovery*

Fase discovery merupakan tahapan awal di mana pengguna baru memasuki sistem aplikasi dan pengenalan aplikasi.

2. Fase *Onboarding*

Fase Onboarding adalah fase di mana pengguna mulai mengenal alur dan aturan aplikasi.

3. Fase *Scaffolding*

Fase ini merupakan fase di mana pengguna mulai menggunakan aplikasi setelah mengenal alur dan misi utama aplikasi.

4. Fase *Endgame*

Fase yang terakhir adalah fase Endgame. Fase ini bertujuan mempertahankan pemain untuk tetap menggunakan aplikasi selepas goal dari aplikasi telah tercapai.

2.2.6.1 *Game mechanics*

Game mechanics merupakan komponen yang diambil dari konsep *game* dalam penerapan gamifikasi. *Game mechanics* diantaranya adalah :

A. Point

Point merupakan alat tukar yang dimiliki pengguna yang berlaku didalam aplikasi yang di gamifikasi. *Point* adalah motivator yang efektif yang dapat digunakan sebagai *reward*/penghargaan ke pengguna, sebagai indikator status, dan sebagai alat tukar pembelian barang atau jasa yang diatur oleh *project manager* atau organisasi [16].

B. Experience point

Experience point adalah poin yang terus bertambah seiring diselesaikannya suatu aktivitas atau pekerjaan dalam sistem. *Experience point* merupakan poin yang paling penting dari berbagai macam jenis poin. Jenis *point* ini digunakan untuk melihat urutan pengguna. *Experience point* pada umumnya bersifat tidak terbatas, tidak bisa ditukarkan, dan tidak dapat kadaluarsa. Tetapi pada beberapa kasus, *experience point* dapat kadaluarsa dengan tujuan untuk menyamakan kembali kedudukan para pengguna [14].

Experience point biasanya digunakan untuk *role-play game* dimana pada awalnya pengguna tidak berpengalaman. Dengan setiap tugas yang diselesaikan, pengguna mendapat pengalaman berupa *experience point* dan siap untuk menghadapi tugas yang lebih sulit atau rumit [17].

C. Redeemable point

Redeemable point adalah poin yang dapat ditukarkan dengan sesuatu. *Redeemable point* bersifat fluktuatif atau nilainya dapat berkurang atau bertambah. Umumnya membentuk pondasi seperti ekonomi *virtual* dan sering diberi nama *coins, bucks, cash*, dan lain-lain. Seperti halnya jenis ekonomi lain, *reward point* harus diawasi dan dikelola perputaran nilainya agar memastikan semuanya berjalan dengan baik serta untuk menghindari inflasi atau deflasi [14].

Cara mudah menentukan nilai barang untuk transaksi dalam dollar adalah 1% dari jumlah *redeemable point*s. Hal ini telah menjadi standar untuk *loyalty program* dan sudah terbukti efektif [11].

D. Level

Level adalah suatu angka yang merepresentasikan seluruh pengalaman. *Level* berfungsi sebagai tanda agar pengguna mengetahui seberapa besar pencapaian dari hasil upaya atau kerja keras yang telah dilakukan. Dalam *game design*, terdapat karakteristik dari sebuah *level* yaitu kalkulasi atau penghitungan *level* selalu memakai fungsi eksponensial. Artinya semakin besar *level* maka semakin besar juga usaha atau kerja keras yang harus dilakukan.

E. Reward

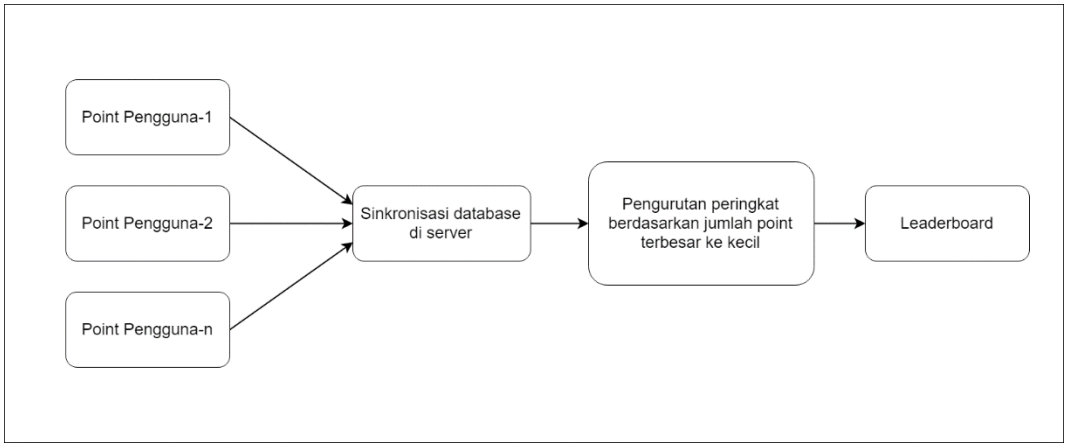
Reward merupakan hadiah yang didapat ketika pengguna menukarkan *point* yang dimilikinya. *Reward* tidak berbentuk uang. Agar pengguna tetap ingin menghabiskan *point*-nya atau tetap mengerjakan aktivitas atau pekerjaannya, *reward* harus terus diperbaharui sesuai kebutuhan. Contoh *reward* yang diterapkan pada lingkungan kerja adalah makan malam, *voucher* belanja, atau *voucher* liburan. [16].

F. Leaderboard

Pengakuan per*form*a antar pengguna yang berdasarkan dari perolehan *point* merupakan hal yang penting. Salah satu cara untuk memunculkan pengakuan ini adalah melalui fitur *leaderboard*. *leaderboard* berisi in*form*asi *ranking* pengguna berdasarkan akumulasi *point* yang dimiliki setiap pengguna yang dapat dilihat oleh pengguna lain. Pengguna yang menempati posisi teratas *leaderboard* mendapatkan manfaat berupa pengakuan, status sosial, dan harga diri. Sedangkan bagi pengguna yang berada pada urutan dibawahnya, timbul keinginanan untuk memperbaiki posisinya. Elemen kompetisi ini yang menjadi penggerak orang-orang, untuk memenuhi kebutuhan mereka akan tantangan dan penghargaan, dan memotivasi daya kerja menuju target.

Untuk beberapa anggota dalam tim, elemen dari pengakuan sosial dapat menjadi lebih penting daripada penghargaan fisik. Para individu yang mencari pengembangan karir dapat lebih menguntungkan dengan pengakuan yang diraih melalui gamifikasi. Rekan-rekannya juga dapat melihat progress masing-masing dan menyadari tugas yang telah dikerjakan secara sempurna, sementara *manager*

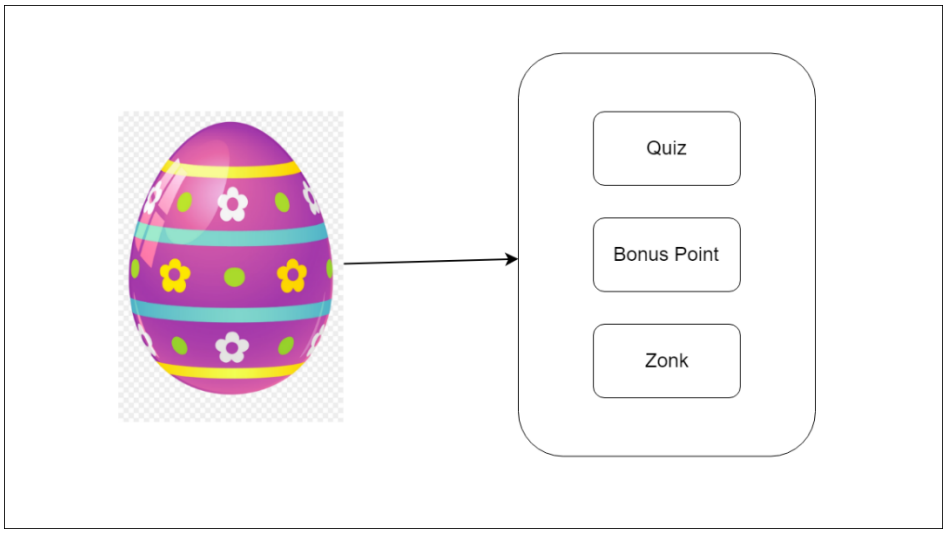
mempunyai kesempatan untuk melihat per*forma* tim [16]. Alur *leaderboard* dapat di lihat pada Gambar berikut.



Gambar 2.14 Alur *Leaderboard*

G. Easter egg

Easter egg merupakan *bonus* tersembunyi bagi pengguna. *Easter egg* dapat berupa tambahan *point* atau lain sebagainya [18]. *Bonus* perlu ada dalam sebuah gamifikasi agar pengguna bersemangat dalam menyelesaikan tugasnya.



Gambar 2.15 Konsep *Easter egg*

2.2.6.2 Manfaat Gamifikasi

Berikut merupakan manfaat utama yang dapat ditimbulkan oleh gamifikasi [16]:

1) Meningkatkan keterlibatan atau partisipasi pengguna.
2) Meningkatkan tingkat motivasi pengguna.
3) Meningkatkan interaksi dengan pengguna (klien atau karyawan).
4) Meningkatkan loyalitas pengguna.

2.2.6.3 Gamifikasi Pada Lingkungan Kerja

Pada lingkungan kerja, motivasi kerja merupakan hal yang harus dimiliki oleh setiap perkerja/pegawai agar mereka dapat memberikan kontribusi positif bagi keberlangsungan perusahaan. Banyak perusahaan yang mengeluarkan banyak uang untuk mengadakan seminar motivasi agar pekerja tidak kehilangan motivasi dalam bekerja [14]. Terdapat beberapa faktor yang menyebabkan pegawai kehilangan motivasi kerja, salah satunya adalah kurangnya pemberian motivasi berupa penghargaan kepada pegawai [14].

Di era komputerisasi ini, banyak *software* yang dibuat untuk membantu proses-proses yang terdapat pada perusahaan sebagai contoh, seperti *software* untuk pengolahan kata/dokumen, penggajian, absensi, hingga *software* yang bermanfaat untuk membantu pengarsipan dan koordinasi kerja antar pegawai atau tim yaitu *software project management*. Salah satu solusi dalam bentuk aplikasi yang membantu/mengurangi *effort* yang harus dikeluarkan pada proses pemberian motivasi kepada team member, adalah dengan menggunakan aplikasi *project management* yang telah di gamifikasi. Saat ini, sudah ada *software Project management* yang mengimplementasikan gamifikasi, yaitu RedCritterTracker. *Game mechanics* yang diterapkan pada aplikas tersebut diantaranya *reward point, lifetime point, badges, rewards, dan leaderboard*. *Game mechanics* tersebut di impelementasikan melalui Fitur-fitur seperti mendapatkan *reward point* dan *lifetime point* saat selesai menyelesaikan suatu tugas, pembelian *reward* menggunakan *reward point*, mendapatkan *badge* saat memenuhi kondisi tertentu,

dan melihat *leaderboard* untuk mengetahui urutan berdasarkan *lifetime point* suatu *user* terhadap *user* lain.

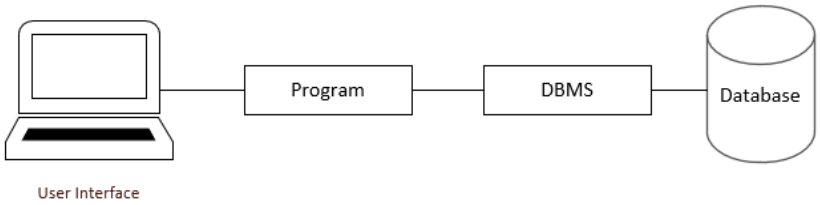
2.2.7 Basis Data

Basis Data terdiri dari dua kata, yaitu Basis dan Data. Basis dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili seuatu objek seperti manusia (pegawai, siswa, pembeli pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya [19].

Sebagai satu kesatuan istilah, basis data dapat diartikan didefinisikan sebagai kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil, dan dicari secara cepat [20].

2.2.7.1 *Database Management System* (DBMS)

Database berbeda dengan *Database Managenent System (DBMS)*. DBMS adalah kumpulan program yang digunakan untuk mendefinisikan, mengatur, dan memproses *database*; sedangkan database itu sendiri esensinya adalah sebuah struktur yang dibangun untuk keperluan penyimpanan data. DBMS alat yang berperan untuk membangun struktur tersebut [20]. Jadi dapat diartikan bahwa DBMS merupakan perantara antara *user* dengan *database*. Peranan DBMS pada suatu sistem dapat dilihat pada gambar 2.4 berikut.



Gambar 2.16 Peranan DBMS dalam Sistem

2.2.7.2 Bahasa Basis Data

Cara berinteraksi antara pemakai dengan basis data diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat DBMS. Bahasa itu dapat kita sebut sebagai Bahasa Basis Data yang terdiri atas sejumlah perintah (statement) yang di*form*ulasikan dan dapat diberikan *user* dan dikenali/diproses oleh DBMS untuk melakukan suatu aksi tertentu. Bahasa Basis Data dapat dibedakan kedalam dua bentuk yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

a) *Data Definition Language* (DDL)

DDL merupakan struktur basis data yang menggambarkan skema basis data secara keseluruhan dan didesain dengan bahasa khusus. Adapun perintah-perintah yang dapat dilakukan dengan DDL yaitu :

1) Membuat, mengubah, menghapus tabel baru.
2) Membuat indeks.
3) Menentukan struktur penyimpanan tabel.
4) Dan sebagainya.

b) *Data Manipulation Language* (DML)

DML merupakan bentuk Bahasa Basis Data yang berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data. Manipulasi data dapat berupa :

1) Penambahan data baru ke suatu basis data
2) Penghapusan data dari suatu basis data
3) Pengubahan data di suatu basis data
4) Pengambilan data dari suatu basis data

2.2.8 *Object Oriented Analysis dan Design* (OOAD)

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). Analisis berorientasi objek (OOA) adalah tahapan menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek. Sedangkan desain berorientasi objek (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek. OOA dan OOD dalam proses yang berulang-ulang sering kali memiliki batasan yang samar, sehingga kedua tahapan ini sering juga disebut Analisis dan Desain Berorientasi Objek (OOAD) [21].

2.2.8.1 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [21]. Dalam pengembangan suatu perangkat lunak, UML digunakan untuk memodelkan suatu sistem yang menggunakan konsep berorientasi object agar lebih bisa dipahami oleh banyak pihak yang terlibat dalam pengembangan.

Terdapat beberapa *diagram* yang biasanya digunakan untuk memodelkan analisis fungsional dalam rangka pengembangan perangkat lunak. Berikut diantaranya *diagram* yang umum digunakan :

a) Use Case Diagram

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol*. *Use case* digambarkan hanya yang dilihat dari luar oleh actor dan bukan bagaimana fungsi yang ada di dalam sistem.

b) Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi. *Activity diagram* dibuat sebanyak aktivitas yang digambarkan pada *use case diagram*.

c) Class Diagram

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram*.

d) *Sequence Diagram*

Menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

2.2.9 Pengujian Perangkat Lunak

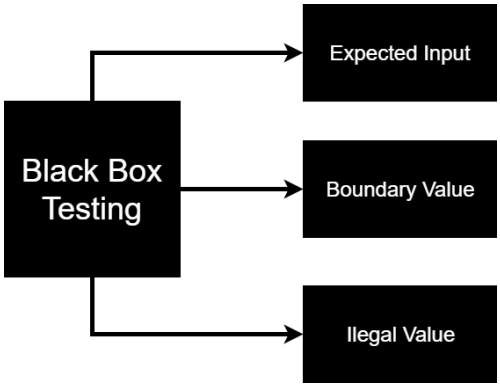
Metode pengujian merupakan metode – metode yang digunakan dalam pengujian perangkat lunak yang dibangun. Pengujian dilakukan untuk memberikan evaluasi atas perangkat lunak yang dibangun, apakah telah sesuai dengan kebutuhan yang menjadi masalah atau belum sesuai.

Berikut merupakan beberapa metode pengujian yang dilakukan pada penelitian ini.

2.2.9.1 Pengujian *Alpha*

Pengujian *alpha* merupakan salah satu strategi pengujian perangkat lunak yang paling umum digunakan dalam pengembangan perangkat lunak. Pengujian ini dilakukan pada sisi pengembang perangkat lunak. Pengujian *alpha* merupakan pengujian akhir sebelum nantinya perangkat lunak di publikasi ataupun digunakan oleh pengguna luas. Metode yang digunakan adalah *Black Box Testing* yaitu pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan

kode program [21]. Metode ini dimaksudkan untuk memastikan semua fungsionalitas berjalan dengan baik dan sesuai dengan apa yang telah direncanakan.



Gambar 2.17 Konsep *Black Box Testing*

Metode *Black Box* memiliki keuntungan dan kekurangan diantaranya :

A. Keuntungan
   1. Efisien diterapkan pada segmentasi kode yang besar.
   2. Persepsi yang harus dimiliki tester sederhana.
   3. Perspektif pengguna dipisahkan dari perspektif pengembang (*programmer* dan tester independen satu sama lain).
   4. Pengembangan kasus uji relatif cepat.

B. Kekurangan
   1. Hanya sejumlah skenario yang dilakukan dan dipilih dalam pengujian. Akibatnya, cakupan pada pengujian terbatas.
   2. Tanpa spesifikasi yang jelas sehingga kasus uji sulit untuk dirancang.
   3. Pengujian tidak efisien.

2.2.9.2 Pengujian *Beta*

Pengujian Beta dilakukan menggunakan kuesioner. Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian responden diminta untuk menilai pernyataan-pernyataan tersebut [22].

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (STS), Tidak Setuju (TS), Ragu-Ragu (R), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel 2.2.

Tabel 2.1 Bobot Pemberian Skor

Jenis pertanyaan	Bobot Pendapat				
	SS	S	R	TS	STS
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu, totalkan seluruh bobot jawaban tersebut kemudian bagi dengan total responden yang nantinya menjadi nilai rata-rata. Nilai rata-rata inilah yang diambil sebagai acuan sikap dimana jika nilai rata-rata kurang dari 3, maka dapat diartikan responden bersikap negatif dan jika nilai rata-rata lebih dari sama dengan 3, maka dapat diartikan responden bersikap positif terhadap tujuan yang ingin dicapai. Untuk lebih jelasnya dapat dilihat pada rumus dibawah ini [23].

$$x = \frac{\sum Total}{n}$$

Dimana

$$x \geq 3 \; bersikap \; positif$$

$$x \leq 3 \; bersikap \; negatif$$

Keterangan

x = nilai rata-rata

$\sum Total$ = jumlah seluruh nilai setelah dikalikan dengan bobot

n = total responden

2.2.10 Perangkat Lunak Pendukung

Perangkat lunak pendukung merupakan perangkat berupa bahasa pemrograman, aplikasi, *framework*, dan sebagainya; yang digunakan untuk mendukung proses pengembangan sistem. Berikut merupakan beberapa perangkat lunak pendukung dalam penelitian ini.

2.2.10.1 Java

Java merupakan bahasa pemrograman berorientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi [24]. Java merupakan bahasa pemrograman yang bersifat *multi-platform* karena program yang dibangun menggunakan bahasa java semuanya berjalan di Java *Runtime Enviroment* (JRE).

Sebelum menjalankan program di bahasa Java, program dikompilasi menggunakan *Java Compiler* [24]. Kompilasi akan menghasilkan *file* "*bytecode*" yang serupa fungsinya dengan *file* kode mesin. Program "*filecode*" yang dihasilkan dapat dieksekusi di sembarang Java *interpreter*. Java *interpreter* membaca file "*bytecode*" dan menterjemahkan perintah "*bytecode*" menjadi perintah-perintah bahasa mesin yang dapat dieksekusi mesin.

2.2.10.2 Android Studio

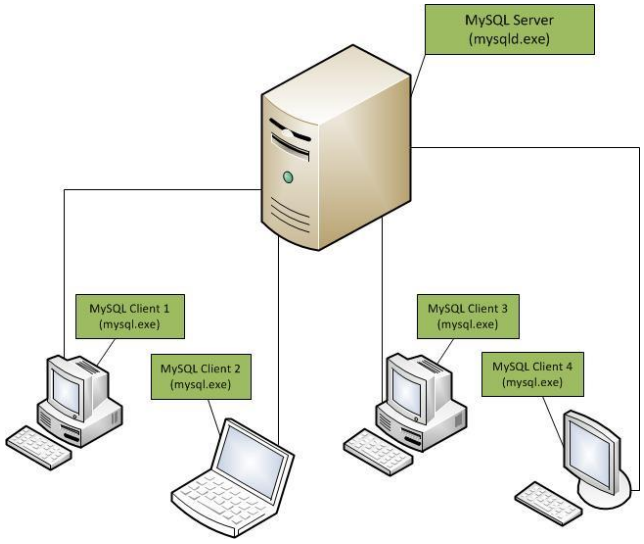
Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berbasis IntelliJ IDEA. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya [25]:

1) Sistem versi berbasis *Gradle* yang fleksibel.
2) Emulator yang cepat dan kaya fitur.
3) Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
4) *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.

5) *Template* kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6) Alat pengujian dan kerangka kerja yang ekstensif.
7) Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
8) Dukungan C++ dan NDK.
9) Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

2.2.10.3 MySQL

MySQL merupakan *software* RDMS (atau *server database*) yang dapat mengelola database dengan sangat cepat, dapat menampung data dalam jumlah besar, dapat diakses oleh banyak *user*, dan dapat melakukan suatu proses secara sinkron atau berbarengan [26]. MySQL merupakan salah satu *Relational Database Management System* (RDBMS) yang paling banyak dipakai oleh para pengembang aplikasi database.



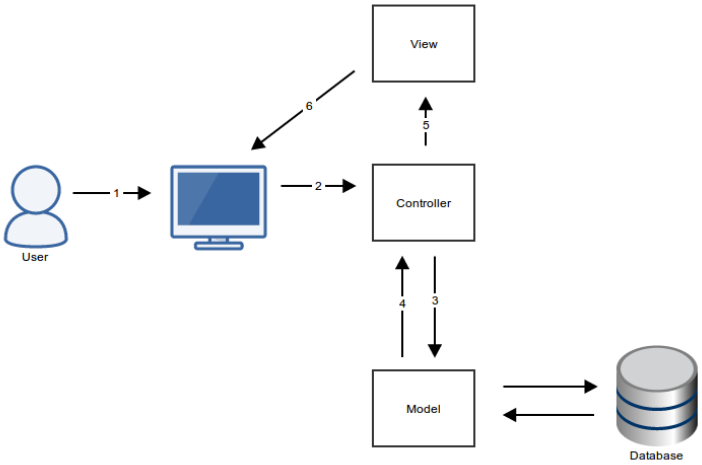
Gambar 2.18 Arsitektur MySQL

2.2.10.4 Laravel

Laravel adalah sebuah *Framework* PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (*model view controller*). Laravel adalah pengembangan *website* berbasis MVC yang ditulis dalam PHP serta dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya

pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

Dimana MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen-komponen aplikasi, seperti : manipulasi data, *controller*, dan *user interface*. Laravel memiliki keunggulan dukungan paket *Library* yang banyak, selain itu Laravel juga menyediakan generator yang canggih dan memudahkan, yaitu artisan dan CLI [27].



Gambar 2.19 Alur Kerja MVC pada Laravel