**BAB II**

**TINJAUAN PUSTAKA DAN LANDASAN TEORI**

**2.1 Tinjauan Pustaka**

Untuk mendukung penelitian ini diperlukan tinjauan pustaka yang diambil dari beberapa jurnal penelitian yang berkaitan dengan judul penelitian ini dan pokok bahasan.

**2.1.1 Perancangan aplikasi media sosial *traveler* berbasis *Gamification* menggunakan google *map* api v3.** **(Fahmi Sulaiman,2013)**

Perkembangan teknologi informasi telah berdampak luas salah satunya berdampak secara signifikan kedalam gaya hidup seseorang. Salah satu dampak dari berkembanganya teknologi informasi adalah aktivitas traveling. Perkembangan akitivitas traveling per tahun dewasa ini terus meningkat secara signifikan. Tingginya tingkat aktivitas traveling membuat aktivitas traveling menjadi sebuah trend di bebagai kalangan. Dengan masalah yang ada yaitu Pencarian informasi suatu objek wisata hanya dapat menggandalkan mesin pencarian secara terpisah melalui portal independent. Berdasarkan masalah yang ada maksud dari penelitian ini adalah untuk Penelitian ini ditujukan untuk membuat rancangan aplikasi media sosial traveler berbasis gamifikasi menggunakan google map api v3. Sejalan dengan tujuan yang dilakukan maka metode yang digunakan adalah metode deskriptif. Penelitian dengan metode deskriptif adalah suatu metode dengan cara mengumpulkan data kemudian disusun dan dianalisis untuk memperoleh gambaran mengenai masalah yang dihadapi pada saat penelitian.

Melalui sistem tersebut Menyajikan media interaktif dan portal informasi bagi para traveler dalam mempermudah ketika melakukan aktivitas traveling. Dengan merancang aplikasi media sosial berbasis web untuk para traveler memungkinkan para traveler dapat dengan mudah berinteraksi secara interaktif dengan objek wisata ataupun mencari informasi mengenai objek wisata yang ada didaerah jawa barat.

Penelitian ini ditujukan untuk membuat rancangan aplikasi media social traveler berbasis gamification menggunakan google map api v3. Sejalan dengan tujuan yang dilakukan maka metode yang digunakan adalah metode deskriptif. Penelitian dengan metode deskriptif adalah suatu metode dengan cara mengumpulkan data kemudian disusun dan dianalisis untuk memperoleh gambaran mengenai masalah yang dihadapi pada saat penelitian.

Hasil dari penlitian pada jurnal ini adalah Aplikasi ini merupakan aplikasi informasi seputar keparawisataan dengan harapan dengan adanya aplikasi ini kita dapat mempelajarai kebudayaan baik itu kebudayaan nasional ataupun kebudayaan internasional, dan Dengan adanya aplikasi ini diharapkan dapat membuat perjalanan traveling lebih menyenangkan serta diharapkan dapat membuat komunikasi dengan media sosial lebih interaktif.

**2.1.2 *Designing gamification for taxi booking system.* (Supriyanto, Hilwadi Hindersah, dkk, 2015)**

Kemacetan lalu lintas pada dasarnya disebabkan oleh kapasitas jalan tidak bisa lagi menampung jumlah kendaraan yang melintas. Selain itu, ada faktor lain yang menyebabkan kemacetan lalu lintas yang terus-menerus terjadi dan lebih sulit untuk menyelesaikan. Faktor-faktor ini termasuk meningkatnya kecenderungan orang untuk menggunakan kendaraan pribadi karena masalah keamanan, kenyamanan dan fleksibilitas transportasi umum. Fenomena ini dapat terus meningkatkan jumlah kendaraan, sehingga semakin sulit untuk mengontrol dan manajemen lalu lintas. Peningkatan jumlah pengguna jalan juga memungkinkan peningkatan pelanggaran lalu lintas dan kejahatan di jalan. Sebuah studi pada transportasi menyebutkan bahwa pelanggaran lalu lintas yang menyebabkan lebih banyak kecelakaan yang dilakukan atau disebabkan oleh driver yang tidak profesional daripada pengemudi profesional, yaitu taksi dan supir angkutan umum.

Dengan adanya masalah pada penelitian ini Oleh karena itu diperlukan suatu inovasi adalah penerapan gamification dalam pelayanan. Tujuannya adalah untuk mendorong tertentu perilaku dan memotivasi para pengguna sistem, dalam hal ini akan digunakan untuk mendorong orang untuk menggunakan publik transportasi melalui taksi pemesanan layanan. Taksi pemesanan layanan terintegrasi dengan aplikasi media sosial berbasis lokasi. Ini memungkinkan layanan taksi dapat digunakan dengan mudah, cepat, dan menyenangkan.

Hasil dari penelitian ini adalah pengujian sistem melibatkan masyarakat pada kota yang diujikan . Responden dipilih secara acak oleh berbagai profesi. Responden yang terlibat adalah semua pengguna jalan, apakah menggunakan kendaraan pribadi atau angkutan umum. Sopir taksi yang dimainkan oleh dua orang dan operasi simulasi di kota yang diuji. simulasi kecil dengan peran dua pengemudi dibuat untuk Semut Taxi tidak dapat didistribusikan kepada supir taksi. Pengujian gamification bertujuan untuk mengetahui pendapat dan tayangan dari responden yang berhubungan dengan penelitian tersebut.

**2.2 Landasan Teori**

Dalam mandukung penelitian ini digunakan beberapa landasan teori yang sesuai dan berkaitan dengan pokok bahasan sebagai berikut :

**2.2.1 Definisi Perancangan**

Perancangan merupakan tahap selanjutnya setelah melakukan analisa terhadap kebutuhan, mendapatkan gambaran dengan jelas tentang apa yang dikerjakan pada analisa kebutuhan, maka dilanjutkan dengan memikirkan bagaimana membentuk sistem tersebut. Perancangan sistem adalah suatu fase dimana diperlukan suatu keahlian perancangan untuk elemen-elemen komputer yang akan mengunakan sistem yaitu pemilihan peralatan dan program komputer untuk sistem yang baru. Adapun tujuan yang hendak dicapai dari tahap perancangan system mempunyai maksud atau tujuan utama, yaitu sebagai berikut:

1. Untuk memenuhi kebutuhan pemakaian sistem (*user*)
2. Untuk memberikan gambaran yang jelas dan menghasilkan rancangan bangun

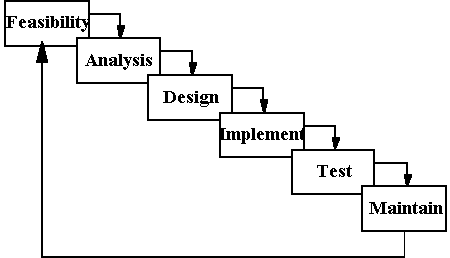
yang lengkap kepada pemograman komputer dan ahli-ahli teknik lainnya yang terlibat dalam pengembangan atau pembuatan sistem.

**2.2.2 Definisi Aplikasi**

Perangkat lunak aplikasi (bahasa Inggris: *software application*) adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media.Aplikasi dapat digolongkan menjadi beberapa kelas, antara lain:

1. Perangkat lunak perusahaan (*enterprise*)
2. Perangkat lunak infrastruktur perusahaan
3. Perangkat lunak informasi kerja
4. Perangkat lunak media dan hiburan
5. Perangkat lunak pendidikan
6. Perangkat lunak pengembangan media
7. Perangkat lunak rekayasa produk.(http://id.wikipedia.org/wiki/Aplikasi)

Siklus hidup pengembangan sistem (SDLC), yang juga disebut sebagai siklus hidup pengembangan aplikasi, digunakan dalam rekayasa sistem, sistem informasi dan rekayasa perangkat lunak, dan menjelaskan suatu prosesdalam pembuatan atau perubahan sistem informasi, dan merupakan model serta metodologi yang digunakan seseorang dalam pengembangan sistem.



Gambar 2.1 Siklus Hidup Perangkat Lunak

**2.2.3 Definisi Android**

Android adalah sebuah kumpulan perangkat lunak untuk perangkat mobile yang mencakup sistem operasi, middleware dan aplikasi utama mobile.

Android memiliki empat karakteristik sebagai berikut:

**1. Terbuka**

Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera, dan lain-lain. Android menggunakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. Android merupakan open source, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. Platform ini akan terus berkembang untuk membangun aplikasi mobile yang inovatif.

**2. Semua aplikasi dibuat sama**

Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (third-party application). Semua aplikasi dapat dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

**3. Memecahkan hambatan pada aplikasi**

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dapat menggabungkan informasi yang diperoleh dari web dengan data pada ponsel seseorang seperti kontak pengguna, kalender, atau lokasi geografis.

**4. Pengembangan aplikasi yang cepat dan mudah**

Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan library yang diperlukan dan tools yang dapat digunakan untuk membangun aplikasi yang semakin baik. Android memiliki sekumpulan tools yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat.

(Sumber : http://www.android.com/about/)

**2.2.4 Definisi Database**

Menurut C.J. Date Database ialah koleksi “data operasional” yang tersimpan dan juga dipakai oleh sistem aplikasi dari suatu organisasi.

* Data input ialah data yang masuk dari luar sistem
* Data output ialah data yang dihasilkan oleh  sistem
* Data operasional ialah data yang tersimpan pada sistem

**a. Xampp**

Menurut Yogi wicaksono (2008:7) “XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolah data *MySQL* dikomputer local”.  XAMPP berperan sebagai *server web* pada komputer anda. XAMPP juga dapat disebut sebuah *CPanel server virtual*, yang dapat membantu anda melakukan *preview* sehingga dapat memodifikasi *website* tanpa harus *online* atau terakses dengan internet.

**b. My Sql**

Menurut Anhar (2010:45) “*MySQL* adalah salah satu *databases* *management system* (DBMS) dari sekian banyak *DBMS* seperti *Oracle, MS SQL, Postagre SQL*, dan lainya”. *MySQL* berfungsi untuk mengolah *database* menggunakan bahasa *SQL. MySQL* bersifat *open source* sehingga kita bisa menggunakannya secara gratis. Pemograman *PHP*  juga sangat mendukung/ *support* dengan *database MySQL*.

**2.2.5 Definisi Gamifikasi**

Menurut badgeville.com, Gamifikasi adalah penggunaan dari teknik desain permainan, permainan berpikir dan permainan mekanik untuk meningkatkan non-game konteks. Biasanya gamifikasi berlaku untuk non-game aplikasi dan proses, untuk mendorong orang untuk mengadopsi mereka, atau untuk mempengaruhi bagaimana mereka digunakan.

Umumnya, gamification mengadopsi salah satu atau beberapa game mechanics yang telah ada, seperti:

1. Badge, merupakan sebuah lencana yang diperoleh user ketika menyelesaikan tugas tertentu.

2. Quest, sebuah misi yang diambil oleh user dan harus dikerjakan untuk mendapatkan point dan reward tertentu.

3. Level, merupakan takaran tinggkatan user dalam permainan untuk menaikan level tertentu user harus mendapatkan point dari quest untuk naik level.

4. Points, merupakan satuan nilai untuk naik ke level selanjutnya.

5. Reward, merupakan hadiah yang akan di dapatkan oleh user ketika menyelesaikan quest tertentu.

6. Experience, merupakan pengalaman user yang di peroleh ketika user menyelesaikan quest yang di nyatakan dalam satuan point.

**2.2.6 UML**

**Menurut Grady Booch, James Rumbaugh, and Ivar Jacobson(1999)***Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor dan metodologi wirfs-brock. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan grup /perusahaan lain yang menggunakan metodologi yang berlainan.

Berikut ini merupakan diagram-diagram yang dapat didefinisikan dari pemodelan UML(http://www.cetus-links.org/oo\_uml.html) :

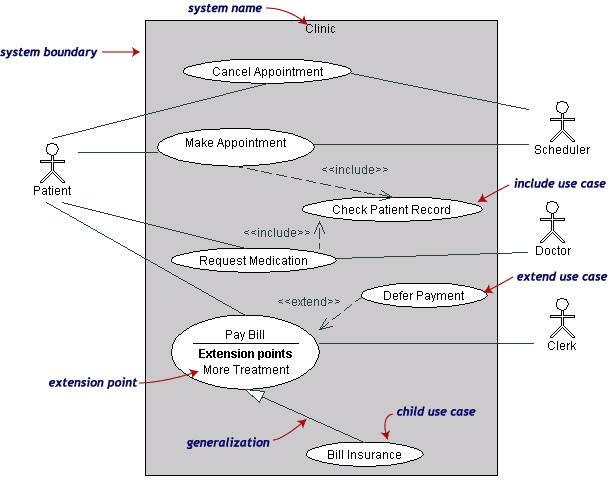
1. ***Use Case Diagram***

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Penekanan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksiantara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua fitur yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai

bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di-*include*akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.Contoh *use case diagram* :



Gambar 2.2 *Use Case Diagram*

1. ***Class Diagram***

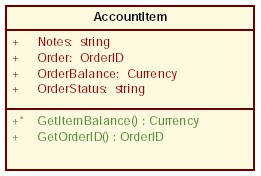
*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

*Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metode

Atribut dan metoda dapat memiliki salah satu sifat berikut :

* *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
* *Protected*, hanya dapat dipanggil oleh *class*yang bersangkutan dan anak-anak yang mewarisinya
* *Public*, dapat dipanggil oleh siapa saja



Gambar 2.3 *Class Diagram*

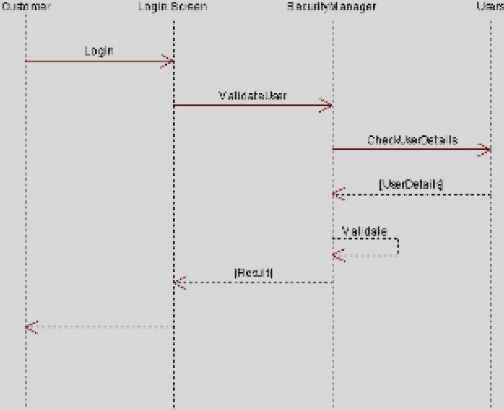
1. ***Sequence Diagram***

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

*Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message*akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary, controller* dan *persistent entity*. Contoh *sequence diagram* :



Gambar 2.4 *Sequence Diagram*

1. ***Activity Diagram***

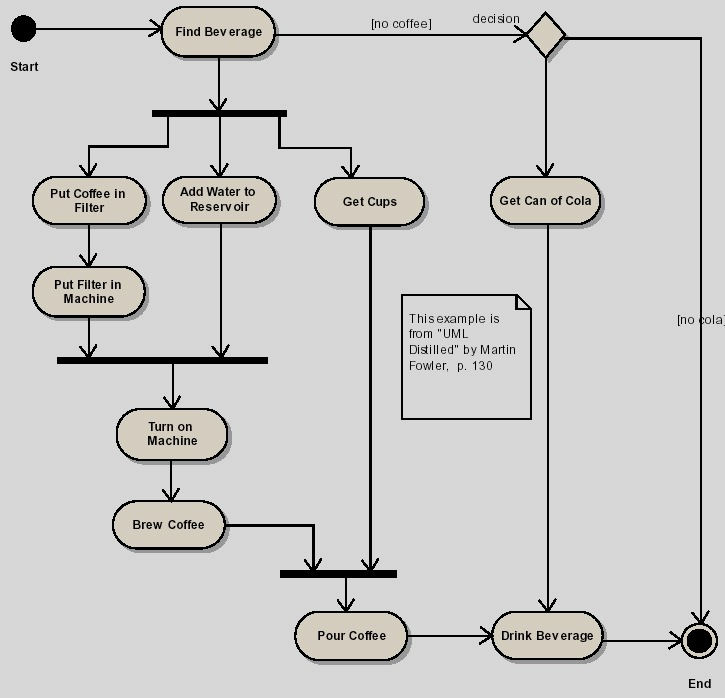
*Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

*Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

*Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh *activity diagram* tanpa*swimlane*:

****

Gambar 2.5 *Activity Diagram*