BAB II
TINJAUAN PUSTAKA

2.1 Aplikasi Mobile

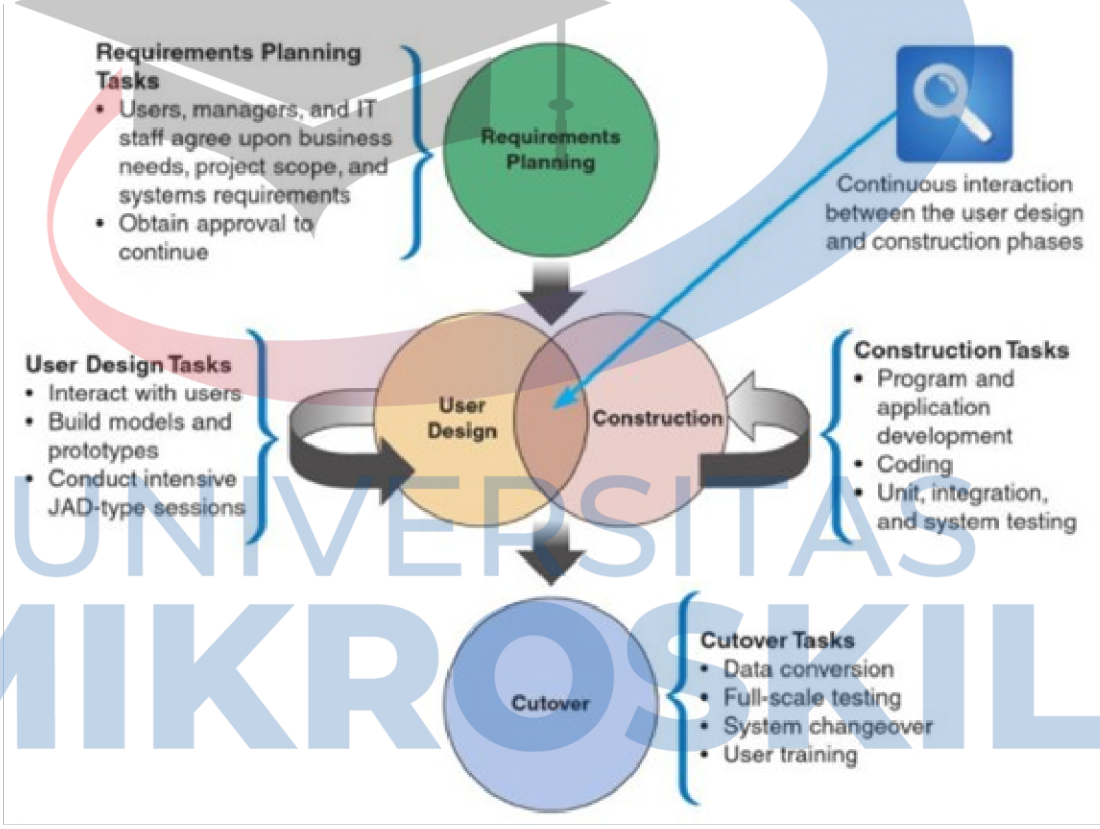
Aplikasi *Mobile* adalah aplikasi yang berjalan pada perangkat seluler seperti ponsel, tablet dan jam tangan. Aplikasi *Mobile* awalnya hanya ditujukan untuk kebutuhan mendasar seperti layanan telepon dan pengiriman pesan. Akan tetapi, perkembangan *mobile computing* yang begitu pesat mengakibatkan aplikasi *mobile* meluas ke sektor lain seperti dunia hiburan sampai ke layanan pemesanan secara daring. Aplikasi *mobile* pada saat ini sudah mampu untuk menyediakan komputasi dan layanan yang lebih rumit seperti *Global Positioning System* (GPS) dan layanan berbasis lokasi [8].

Berdasarkan teknologi yang digunakan, aplikasi *mobile* terbagi kedalam tiga bagian [8]:

1. *Native Application* yang dibuat khusus untuk suatu sistem operasi tertentu. Karena *native application* dibuat secara khusus untuk terhubung dengan sistem operasi tersebut, maka aplikasi jenis ini cenderung memiliki performa yang lebih baik dan memiliki lebih banyak akses terhadap perangkat keras pengguna. Akan tetapi, *Native Application* tidak bisa dijalankan pada sistem operasi lain.
2. *Web Application* yang diakses melalui web dengan perangkat seluler. *Web Application* bukan merupakan suatu aplikasi yang berdiri sendiri untuk perangkat *mobile* melainkan merupakan situs web dengan UI yang responsif terhadap tampilan *mobile*. Karena pada dasarnya *Web Application* diakses melalui *browser* pengguna, maka pengalaman pengguna bisa berbeda-beda tergantung pada *browser* apa yang digunakan.
3. *Hybrid Application* yang merupakan gabungan dari *Web Application* dan *Native Application*. *Hybrid Application* berjalan melalui *browser* akan tetapi bisa mengakses berbagai fitur-fitur yang terdapat pada *Native Application*.

2.2 Rapid Application Development

Rapid Application Development (RAD) adalah metodologi untuk mempercepat pengembangan sistem informasi dan menghasilkan sistem informasi yang fungsional [9]. Tujuan utama dari semua pendekatan RAD adalah untuk memangkas waktu pengembangan. Karena setiap tahapan merupakan proses yang berkelanjutan, RAD memungkinkan tim pengembang untuk melakukan modifikasi yang diperlukan secara cepat ketika terdapat perubahan yang tidak diperkirakan sebelumnya. Tahapan RAD terbagi kedalam 4 tahapan [9], yaitu:



Gambar 2.1 Empat tahapan pada *Rapid Application Development*

1. *Requirements Planning*

Perencanaan kebutuhan menggabungkan elemen-elemen dari tahap perencanaan sistem dan analisis sistem pada metodologi SDLC. Pengguna, manajer dan staf TI

berdiskusi dan menyepakati ruang lingkup, batasan dan kebutuhan sistem. Tahapan ini selesai apabila kesepakatan telah tercapai.

2. *User Design*

Di dalam tahapan desain, pengguna berinteraksi dengan analis sistem dan membangun model dan *prototype* yang mewakili semua masukan, proses dan keluaran sistem. Desain pengguna merupakan proses berkelanjutan dan interaktif yang memberikan pengguna kesempatan untuk memahami, memodifikasi dan menyetujui model sistem yang memenuhi kebutuhan mereka.

3. *Construction Phase*

Tahapan konstruksi berfokus pada pengembangan program dan aplikasi. Akan tetapi pada tahapan ini, perubahan atau perkembangan masih bisa diusulkan dan diterapkan.

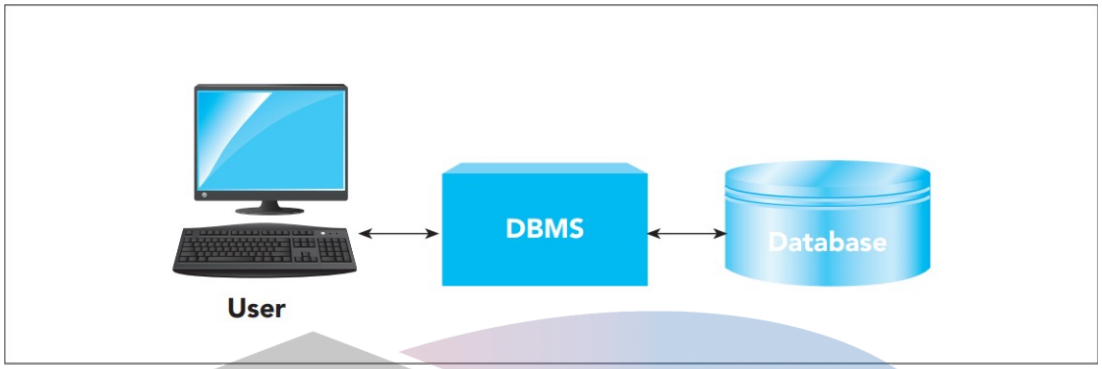
4. *Cutover Phase*

Fase *Cutover* menyerupai tahapan akhir pada fase implementasi di dalam SDLC, seperti konversi data, pengujian dan pelatihan pengguna.

2.3 Basis Data

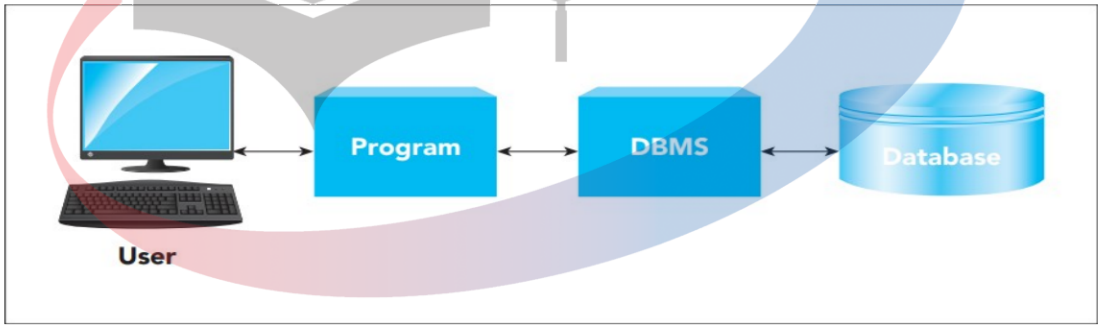
2.3.1 Sistem Manajemen Basis Data

Basis data adalah sekumpulan data atau informasi yang terstruktur, terorganisir dan disimpan secara elektronik. Mengelola basis data adalah tugas yang rumit. Untungnya, terdapat paket perangkat lunak yang disebut Sistem Manajemen Basis Data yang dapat melakukan manipulasi basis data. Sistem Manajemen Basis Data atau *Database Management System (DBMS)* adalah sebuah program, atau kumpulan program, di mana pengguna berinteraksi dengan basis data [10].



Gambar 2.2 Menggunakan DBMS secara langsung.

Meskipun manipulasi basis data ditangani oleh DBMS, pengguna juga bisa berinterakasi secara langsung dengan DBMS seperti yang ditunjukkan pada Gambar 2.5.



Gambar 2.3 Berinteraksi dengan DBMS melalui suatu program.

Dalam kasus lain, pengguna berinteraksi dengan DBMS melalui program perantara yang dibuat dengan *Visual Basic*, *Java*, *Perl*, *PHP, C++* atau bahasa pemrograman lain seperti yang ditunjukkan pada Gambar 2.5. Dalam kedua kasus tersebut, hanya DBMS yang benar-benar mengakses database [10].

Basis data yang didasarkan pada tabel merupakan salah satu jenis basis data yang disebut dengan basis data relasional. Gambar 2.5 memperlihatkan contoh tabel yang mendeskripsikan judul film, tahun, durasi, genre dan bagaimana hubungan antara keempat kolom tersebut. Misalnya, film berjudul *Star Wars* (baris dua) diproduksi pada tahun 1977 dengan durasi 124 menit dan memiliki genre *sciFi*.

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Gambar 2. 4 Contoh tabel relasi.

Secara formal tabel-tabel pada basis data ini disebut relasi dan dikelola oleh Sistem Manajemen Basis Data Relasional atau *Relational Database Management System (RDBMS)*. RDBMS adalah DBMS yang memiliki dukungan terhadap basis data relasional.

Sistem Manajemen Basis Data diharapkan mampu [11]:

1. Mengizinkan pengguna untuk membuat database baru dan menentukan skema basis data (struktur logis data) menggunakan suatu *data-definition language*.
2. Memberi pengguna kemampuan untuk melakukan kueri data dan memodifikasi data menggunakan bahasa yang sesuai, suatu bahasa yang sering disebut bahasa kueri atau *data-manipulation language*.
3. Mendukung penyimpanan data dalam jumlah yang sangat besar dalam jangka waktu yang lama serta memungkinkan akses data yang efisien.
4. Memungkinkan pemulihan database untuk menghadapi kegagalan, berbagai jenis kesalahan, atau penyalahgunaan yang disengaja.
5. Kontrol akses ke data dari banyak pengguna sekaligus tanpa mengizinkan interaksi tak terduga di antara pengguna (disebut *isolation*) dan tanpa mengizinkan tindakan yang dilakukan secara parsial terhadap data (disebut *atomicity*).

2.3.2 Konsep Basis Data Relasional

Sebuah basis data relasional adalah kumpulan dari tabel-tabel yang saling terhubung. Basis data relasional mengelola entitas, atribut dari setiap entitas, dan hubungan antar entitasnya. Setiap entitas harus disimpan memiliki tabelnya masing-masing. Dikatakan berhubungan dikarenakan [10]:

1. Setiap masukkan ke dalam tabel merupakan nilai tunggal yang letaknya pada tabel mewakili satu nilai
2. Setiap kolom memiliki nama yang berbeda (secara teknis disebut nama atribut)
3. Setiap nilai yang ada pada kolom memiliki nilai yang sama pada setiap atributnya
4. Urutan dari kolomnya tidak ada pengaruhnya
5. Setiap baris harus berbeda dari lainnya
6. Urutan dari setiap baris tidak ada pengaruhnya

2.4 Bahasa Pemrograman

2.4.1 Kotlin

Kotlin adalah bahasa pemrograman modern [12] dengan tipe statis yang artinya tipe bahasa pemrograman yang setelah diselesaikan pada waktu kompilasi maka tidak pernah berubah. *Kotlin* juga dikenal dengan kecocokannya terhadap android karena banyak melakukan perbaikan pada kekurangan bahasa pemrograman *java*, seperti *null pointer exceptions* atau *excessive code verbosity*. *Kotlin* terinspirasi dari banyak bahasa pemrograman lain seperti *Swift*, *Scala*, *Groovy*, *C#* dan sebagainya [12].

Kotlin pada tahun 2017 diumumkan secara resmi sebagai rekomendasi bahasa pemrograman untuk *Android*, hal ini membawa *Kotlin* pada tingkat pemgembangan aplikasi *Android* pada tingkat yang lebih baik dari segi keamanan, kualitas koding, percepatan pengembangan aplikasi. Oleh karena itu, *Kotlin* bisa disebut memiliki fitur yang *safe, expressive, concise, versatile dan tool-friendly language* yang bisa berkesinambungan dengan bahasa pemrograman lain seperti *Java* dan *JavaScript*. Adapun penjelasan fitur-fitur *Kotlin* sebagai berikut [12]:

1. *Safety*: *Kotlin* menawarkan fitur *nullability* dan *immutability* yang menjadikan *Kotlin* sebagai bahasa yang ketat. Tanpa fitur ini program yang dikembangkan akan sering *crash*.

2. *Easy debugging*: *Kotlin* mampu membedakan antara mutable (*read-write*) dan *immutable* (read only) sehingga memudahkan para pengembang untuk proses *debug*.
3. *Conciseness*: Dibandingkan dengan *Java* yang terlalu banyak seremoni dalam penulisan kodenya, *Kotlin* hadir dengan cara yang jauh lebih ringkas dalam penulisan kodenya. Hal ini menyebabkan para pengembang dengan mudah dalam membaca dan memahami(ekspresif).
4. *Interoperability*: Aplikasi yang menggunakan bahasa pemrograman *Java* sebelumnya bisa langsung dilanjutkan menggunakan *Kotlin*, hal ini diakibatkan banyaknya *library* dan *Framework Java* yang juga bisa berjalan menggunakan *Kotlin* tanpa harus kehilangan fitur-fitur penting, begitupun sebaliknya.
5. *Versatility*: *Platform* yang luas meliputi aplikasi *mobile* (*Android*), sisi server (*Backend*), *desktop*, *website* (*Frontend*), *Gradle*.

2.4.2 PHP dan Framework Laravel

Laravel adalah kerangka kerja pengembangan *web* yang ditulis dalam bahasa pemrograman PHP. Laravel telah dirancang untuk meningkatkan kualitas perangkat lunak dengan cara mengurangi biaya pengembangan di awal beserta biaya pemeliharaannya, dan untuk meningkatkan pengalaman bekerja dengan menyediakan berbagai sintaks ekspresif, jelas dan menghemat waktu [13].

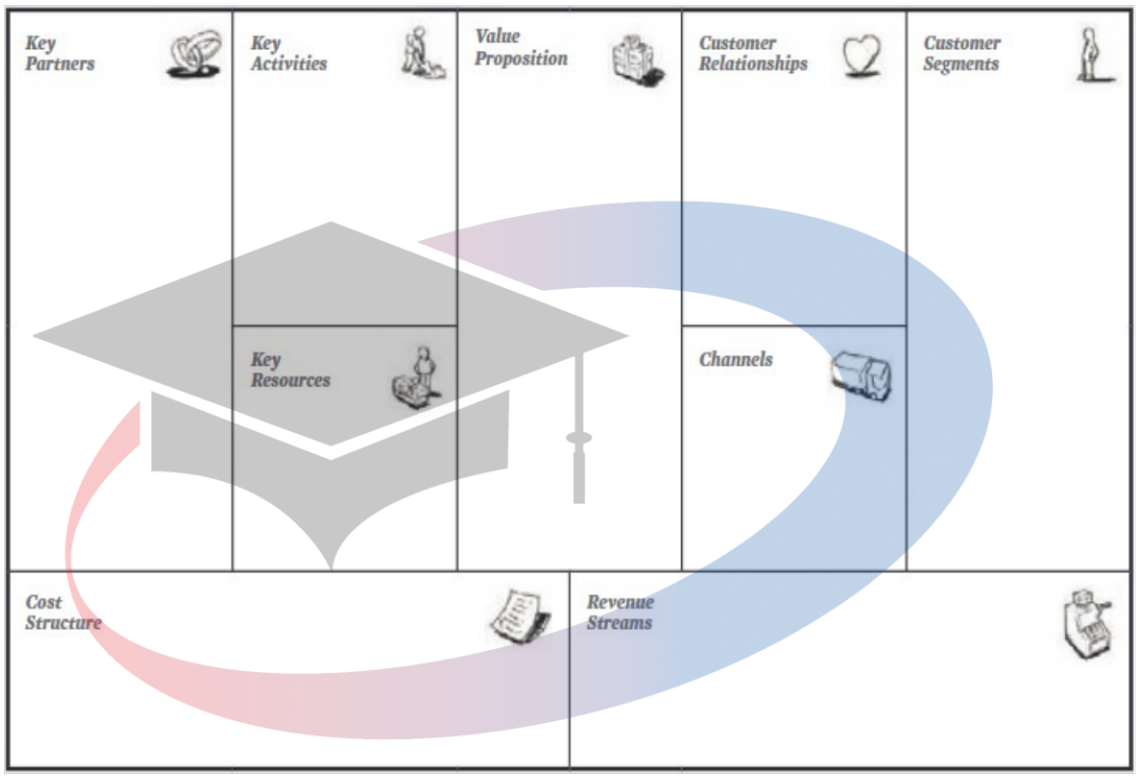
Laravel menyediakan serangkaian alat untuk berinteraksi dengan berbagai basis data (MySQL, PostgreSQL, MSSQL, SQLite) sehingga migrasi dan modifikasi basis data lebih mudah dilakukan. *Laravel Fluent Query Builder* mengabstraksi perbedaan-perbedaan yang ada diantara basis data, sehingga query yang digunakan bisa dilakukan dengan cara yang benar [13].

2.5 Teknik Pengembangan Sistem

2.5.1 Business Model Canvas

Business Model Canvas (BMC) adalah alat manajemen strategis untuk mendefinisikan dan mengomunikasikan ide atau konsep bisnis dengan cepat dan mudah.

BMC adalah dokumen satu halaman yang bekerja melalui elemen dasar bisnis atau produk, menyusun ide dengan cara yang koheren [14].



Gambar 2.5 *Template* Business Model Canvas

Business Model Canvas terbagi menjadi beberapa bagian yaitu:

1. *Customer Segments*

Pada bagian ini didefinisikan sekelompok orang atau organisasi yang hendak diraih dan dilayani oleh perusahaan atau yang biasa disebut dengan pelanggan. Pelanggan merupakan inti dari segala bisnis, tanpa pelanggan tidak ada perusahaan yang akan bertahan dalam menjalani usahanya. Dalam rangka pemenuhan kepuasan pelanggan, perusahaan bisa mengelompokkan para pelanggannya berdasarkan kebutuhannya, kebiasaannya dan atribut lainnya.

Sebuah model bisnis bisa mendefinisikan satu atau beberapa segmen pelanggan, baik dalam ukuran yang besar maupun kecil. Sebuah organisasi harus membuat keputusan secara sadar dalam memilih pelanggan yang akan dilayani atau tidak. Setelah

keputusan tersebut diambil, model bisnis bisa didesain dengan pemahaman yang lebih baik tentang kebutuhan spesifik pelanggan.

2. *Value Propositions*

Pada bagian ini mendeskripsikan sekumpulan produk atau jasa yang dapat menciptakan nilai tambah terhadap *Customer Segments* yang spesifik. *Value Proposition* menjadi salah satu alasan pelanggan bisa berpaling dari satu perusahaan ke perusahaan lainnya karena mungkin perusahaan lain mampu menyelesaikan permasalahan pelanggan secara lebih baik. *Value Proposition* bisa berupa hal yang inovatif dan menggambarkan sesuatu yang baru atau mendisrupsi penawaran yang ada sebelumnya seperti misalnya penambahan fitur atau atribut baru.

3. *Channels*

Pada bagian ini mendeskripsikan bagaimana perusahaan berkomunikasi dan menggapai *Customer Segments* dalam rangka mengirimkan *Value Propositions*. Komunikasi, distribusi, dan saluran penjualan dapat digunakan untuk menggapai pelanggan. *Channels* merupakan *touch point* yang penting dalam membentuk pengalaman pelanggan seperti:

a. Meningkatkan perhatian di antara pelanggan tentang adanya produk atau jasa yang ditawarkan oleh sebuah perusahaan
b. Membantu pelanggan dalam mengevaluasi *Value Proposition* dari sebuah perusahaan
c. Biarkan pelanggan memilih sendiri produk atau jasa yang ditawarkan perusahaan
d. Menyampaikan sebuah *Value Proposition* pada pelanggan
e. Menyediakan layanan bantuan pasca pembelian

4. *Customer Relationships*

Customer Relationships menjelaskan jenis hubungan yang dibangun perusahaan dengan *Customer Segments* tertentu. Perusahaan harus mengklarifikasi jenis hubungan

yang ingin dibangun dengan setiap *Customer Segments*. Hubungan dapat berkisar dari pribadi hingga otomatis seperti misalnya *automation* terhadap produk atau jasa. *Customer Relationships* dapat didorong oleh motivasi berikut: Akuisisi pelanggan, Retensi pelanggan, Meningkatkan penjualan (*upselling*). *Customer Relationships* yang dibutuhkan dalam model bisnis akan sangat mempengaruhi pengalaman pelanggan secara keseluruhan.

5. Revenue Streams

Revenue Streams mewakili uang yang dihasilkan perusahaan dari setiap *Customer Segments* (*Cost Structure* harus dikurangi dari pendapatan untuk menciptakan pendapatan). Jika pelanggan merupakan jantung dari model bisnis maka *Revenue Stream* adalah arterinya. Sebuah perusahaan harus bertanya pada dirinya sendiri, untuk nilai seperti apa sehingga setiap *Customer Segments* benar-benar bersedia membayar? Berhasil menjawab pertanyaan itu memungkinkan perusahaan menghasilkan satu atau lebih *Revenue Stream* dari setiap *Customer Segments*. Setiap *Revenue Stream* mungkin memiliki mekanisme penetapan harga yang berbeda, seperti daftar harga tetap, tawar-menawar, lelang, ketergantungan pasar atau ketergantungan volume. *Revenue Stream* dapat melibatkan dua jenis *Revenue Stream* yang berbeda:

a. Pendapatan transaksi yang dihasilkan dari satu kali pembayaran pelanggan
b. Pendapatan berulang yang dihasilkan dari pembayaran berkelanjutan untuk memberikan Value Proposition kepada pelanggan atau memberikan dukungan pelanggan pasca pembelian.

6. Key Resources

Key Resources menjelaskan aset terpenting yang diperlukan untuk membuat model bisnis berfungsi. Setiap model bisnis memerlukan *Key Resources* yang memungkinkan perusahaan untuk membuat dan menawarkan *Value Proposition*, menjangkau pasar, memelihara hubungan dengan *Customer Segments*, dan memperoleh pendapatan. *Key Resources* yang berbeda diperlukan tergantung pada jenis model bisnis. Produsen

microchip membutuhkan fasilitas produksi yang padat modal, sedangkan perancang *microchip* lebih berfokus pada sumber daya manusia. *Key Resources* dapat berupa fisik, keuangan, intelektual, atau manusia. *Key Resources* dapat dimiliki atau disewa oleh perusahaan atau diperoleh dari mitra utama.

7. *Key Activities*

Key Activities menjelaskan hal terpenting yang harus dilakukan perusahaan untuk membuat model bisnisnya berfungsi. Setiap model bisnis memerlukan sejumlah *Key Activities*. Ini adalah tindakan paling penting yang harus diambil perusahaan untuk beroperasi dengan sukses. Seperti *Key Resources*, mereka diharuskan untuk membuat dan menawarkan *Value Proposition*, menjangkau pasar, memelihara *Customer Relationships*, dan memperoleh pendapatan. Dan seperti *Key Resources, Key Activities* berbeda tergantung pada jenis model bisnis. Untuk pembuat perangkat lunak Microsoft, *Key Activities* meliputi pengembangan perangkat lunak. Untuk produsen PC Dell, *Key Activities* mencakup manajemen rantai pasokan.

8. *Key Partnerships*

Key Partnerships menggambarkan jaringan pemasok dan mitra yang membuat model bisnis berfungsi. Perusahaan menjalin kemitraan karena berbagai alasan, dan kemitraan menjadi landasan dari banyak model bisnis. Perusahaan membuat aliansi untuk mengoptimalkan model bisnis mereka, mengurangi risiko, atau memperoleh sumber daya. Kita dapat membedakan antara empat jenis kemitraan yang berbeda:

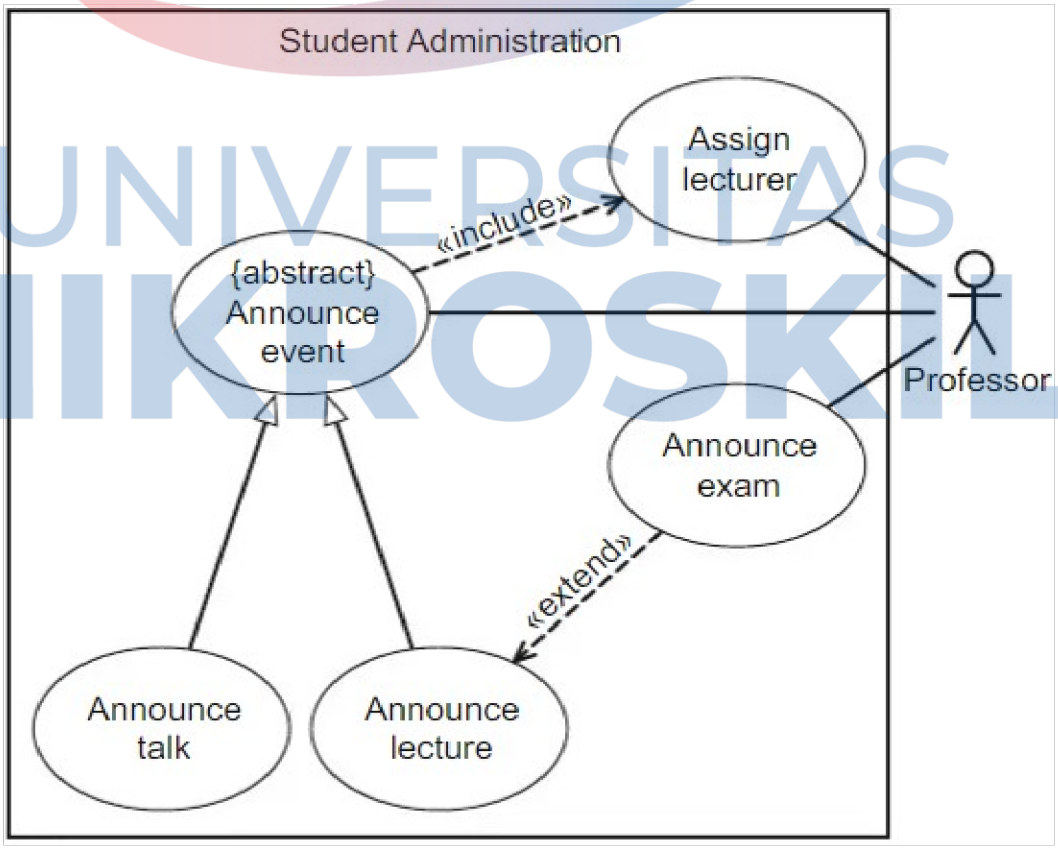
a. Aliansi strategis antara non-pesaing
b. kemitraan strategis antar pesaing
c. Usaha patungan untuk mengembangkan bisnis baru
d. Hubungan pembeli-pemasok untuk memastikan pasokan yang andal

9. *Cost Structure*

Cost Structure menggambarkan semua biaya yang dikeluarkan untuk mengoperasikan model bisnis. Bagian ini menjelaskan biaya terpenting yang dikeluarkan saat beroperasi di bawah model bisnis tertentu. Menciptakan dan memberikan nilai, memelihara *Customer Relationships*, dan menghasilkan pendapatan semuanya memerlukan biaya. Biaya tersebut dapat dihitung dengan relatif mudah setelah menentukan *Key Resources, Key Activities, dan Key Partnerships.*

2.5.2 Use Case Diagram

Use Case Diagram memungkinkan kita untuk menggambarkan kemungkinan skenario penggunaan (*use case*) yang dikembangkan untuk suatu sistem. *Use Case Diagram* mengungkapkan apa yang harus dilakukan sistem tetapi tidak membahas detail dari realisasi apapun seperti struktur data, algoritma, dan sebagainya. *Use Case Diagram* memodelkan fungsionalitas dari pengguna sistem, yaitu mengungkapkan siapa yang benar-benar akan bekerja dengan sistem yang akan dibangun. [15]

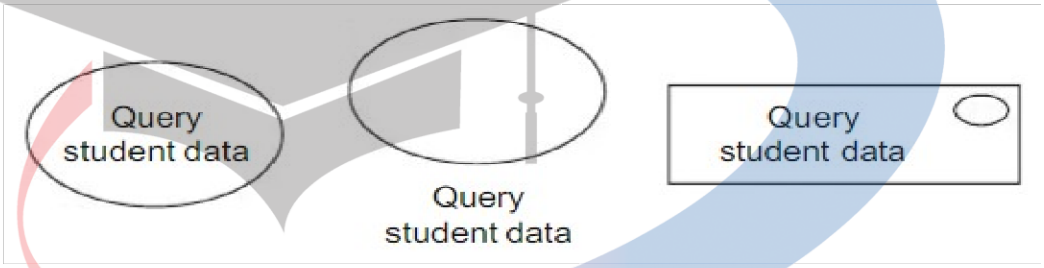


Gambar 2.6 *Use Case Diagram*

Komponen pada *Use Case Diagram* adalah sebagai berikut:

1. Use Case

Sebuah *use case* mendeskripsikan fungsionalitas yang diharapkan dari sebuah sistem yang akan dikembangkan. Ini mencakup sejumlah fungsi yang dijalankan saat menggunakan sistem ini. Sebuah *use case* memberikan manfaat nyata bagi satu atau lebih aktor yang berkomunikasi dengan *use case* ini. Pada umumnya, sebuah *use case* baru dijalankan apabila ada seorang aktor yang memulainya.

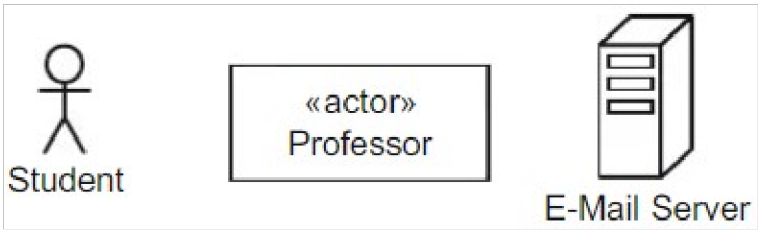


Gambar 2.7 Simbol *use case*

Sebuah *use case* biasanya direpresentasikan sebagai elips. Nama *use case* ditentukan langsung di dalam atau langsung di bawah elips. Sebagai alternatif, sebuah *use case* dapat diwakili oleh sebuah persegi panjang yang berisi nama *use case* di tengah dan sebuah elips kecil di sudut kanan atas.

2. Aktor

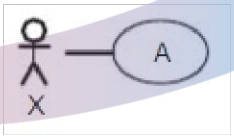
Untuk menggambarkan sistem secara lengkap, penting untuk mendokumentasikan tidak hanya apa yang dapat dilakukan sistem tetapi juga siapa yang benar-benar bekerja dan berinteraksi dengan sistem. Dalam *use case diagram*, aktor selalu berinteraksi dengan sistem dalam konteks use case mereka, yaitu *use case* yang terkait dengan mereka. Aktor diwakili oleh gambar orang-orangan, persegi panjang dengan tambahan informasi <<aktor>> atau simbol lainnya. Aktor bisa berupa manusia (mahasiswa atau dosen) atau non-manusia (email atau server).



Gambar 2.8 Simbol *aktor*

3. Asosiasi

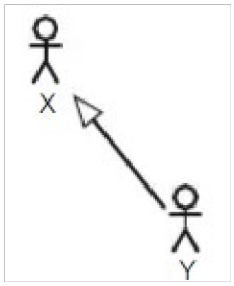
Cara menghubungkan antara aktor dengan *use case* adalah dengan menggunakan garis solid. Seorang aktor yang dihubungkan dengan *use case* menggunakan asosiasi bisa diartikan bahwa aktor sedang berkomunikasi dengan sistem dan menggunakan fungsionalitas tertentu. Setiap aktor harus berkomunikasi dengan paling tidak satu *use case,* begitu pun sebaliknya setiap *use case* harus memiliki asosiasi dengan paling tidak satu aktor.



Gambar 2.9 Simbol *Asosiasi*

4. Hubungan antar aktor

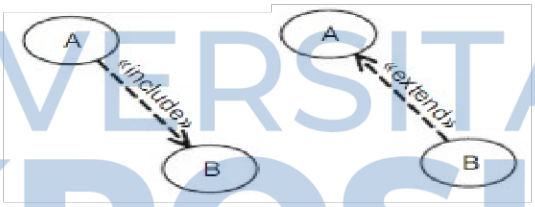
Aktor sering memiliki properti umum dan beberapa kasus penggunaan dapat digunakan oleh berbagai aktor. Misalnya, mungkin tidak hanya profesor tetapi juga asisten (yaitu, seluruh personel penelitian) diizinkan untuk melihat data siswa. Untuk mengungkapkan hal ini, aktor dapat digambarkan dalam hubungan pewarisan (generalisasi) satu sama lain. Ketika aktor Y (sub-aktor) mewarisi dari aktor X (aktor super) maka Y terlibat dengan X. Secara sederhana, generalisasi mengungkapkan hubungan "adalah". Itu diwakili dengan garis dari sub-aktor ke super-aktor dengan panah segitiga besar di ujung aktor super.



Gambar 2.10 Hubungan antar aktor

5. Hubungan antar *use case*

Use case juga bisa berhubungan dengan *use case* lainnya. Di sini kita membedakan antara hubungan *«include»*, hubungan *«extend»*, dan generalisasi kasus penggunaan. Tujuan dari hubungan *«include»* dan *«extend»* adalah untuk melihat kebutuhan dari masing-masing *use case.* Apabila *use case A* dihubungkan dengan *use case B* menggunakan hubungan *«include»,* maka *use case B* harus dijalankan agar *use case A* bisa menjalankan fungsionalitasnya. Namun apabila dihubungkan dengan hubungan *«extend»* maka *use case A* bisa menjalankan fungsionalitas *use case B*, tetapi tidak harus mengambil fungsionalitas *use case B*.



Gambar 2.11 Hubungan antar *use case*

2.5.3 PIECES

PIECES adalah kerangka untuk menganalisis kebutuhan atau permasalahan pada sistem melalui enam aspek sebagai berikut [16]:

1. *Performance*

Kinerja sistem dapat diukur melalui jumlah pekerjaan yang bisa dilakukan oleh sistem dalam kurun waktu tertentu ataupun waktu tanggap sistem.

2. *Information*

a. *Outputs*

i. Kurangnya informasi

ii. Kurang pentingnya informasi

iii. Informasi yang tidak relevan

iv. Informasi yang tidak sesuai format

v. Informasi yang tidak akurat

vi. Informasi yang susah diproduksi

vii. Informasi yang tidak tepat waktu

b. *Inputs*

i. Data yang tidak bisa didapat

ii. Data didapatkan namun diwaktu yang tidak tepat

iii. Data didapatkan dengan cara yang tidak akurat (mengandung kesalahan)

iv. Data yang sulit didapat

v. Data yang didapat redundan

vi. Terlalu banyak data yang didapat

vii. Terlalu banyak data untuk didapat

viii. Data yang didapat ilegal

c. *Stored data*

i. Data disimpan secara redundan

ii. Data yang sama memiliki nilai yang berbeda pada *file* yang berbeda

iii. Data yang disimpan tidak akurat

iv. Data yang tidak aman rawan terjadi kecelakaan

v. Data yang tidak dirapikan dengan baik

vi. Data tidak fleksibel – tidak mudah untuk mendapatkan informasi baru dari data yang tersimpan

vii. Data tidak bisa diakses

3. *Economy*
    a. Biaya
        i. Biaya yang tidak diketahui
        ii. Biaya yang tidak bisa dilacak dari sumbernya
        iii. Biaya terlalu tinggi
    b. Keuntungan
        i. Pangsa pasar baru yang bisa di eksplor
        ii. Pemasaran saat ini yang bisa ditingkatkan
        iii. Pesanan bisa ditingkatkan
4. *Control*
    a. Terlalu sedikit keamanan atau kontrol
        i. Data masukan tidak diedit secara memadai
        ii. Kejahatan yang dilakukan atas data
        iii. Data diakses oleh orang yang tidak memiliki otoritas
        iv. Redundan menyebabkan inkonsisten
        v. Privasi dan regulasi data yang dilanggar
    b. Terlalu banyak control atau keamanan
        i. Birokrasi menyebabkan lambatnya sistem
        ii. Kontrol pada konsumen dan pekerja membuat ketidaknyamanan
        iii. Hambatan proses
5. *Efficiency*
    a. Orang, mesin atau komputer yang bekerja secara tidak efisien
    b. Orang, mesin atau komputer yang menghabiskan sumber daya dan persediaan
6. *Service*
    a. Sistem memproduksi hasil yang tidak akurat
    b. Sistem memproduksi hasil yang tidak konsisten
    c. Sistem memproduksi hasil yang tidak bisa diandalkan

d. Sulit dipelajari

e. Tidak mudah digunakan

2.6 Alat Pendukung Pengembangan Aplikasi

2.6.1 Android Studio

Android Studio adalah IDE (*integrated development environment*) resmi untuk pengembangan Android [17]. Android Studio sudah mencakup berbagai kebutuhan untuk memulai pengembangan sebuah aplikasi Android seperti SDK (*Software Development Kit), Android library* yang akan digunakan, emulator untuk menjalankan aplikasi tanpa memerlukan perangkat nyata.

Keuntungan dari menggunakan Android Studio sebagai IDE dalam mengembangan aplikasi Android adalah sebagai berikut [17]:

1. Pembangunan sistem berbasis *gradle* lebih fleksibel
2. *Template* kode dalam membantu membangun fitur-fitur umum
3. Membangun varian dan pembuatan beberapa *file APK* sekaligus
4. Dukungan bawaan untuk *Google Cloud Platform*
5. Dukungan resmi dari Google sehingga pembaruan tidak memerlukan migrasi

2.6.2 Visual Studio Code

Visual Studio Code (VS Code) adalah kode editor dengan fitur lengkap yang dioptimalkan untuk pengembangan *web*, seluler, dan *cloud.* VS Code mendukung penuh siklus hidup dari sebuah pengembangan aplikasi seperti *debuger* bawaan sampai kontrol versi Git. Dengan VS Code pengembang bisa mengerjakan kode program secara tunggal maupun dengan sistem terstruktur berdasarkan *folder* [18].

Kekuatan VS Code dalam membantu pengembang hanya akan terlihat ketika dihadapkan pada aplikasi asli. Dengan bantuan yang spesifik, VS Code dapat dengan mudah menghasilkan proyek .NET Core menggunakan bahasa pemrograman C# ataupun proyek Node.js. VS Code bahkan bisa melakukan hal yang lebih rumit seperti

memproduksi aplikasi dan fungsi-fungsi dari Azure, mengemas gambar Docker, dan menjalankan jasa kecerdasan buatan [18].

2.6.3 MariaDB

MariaDB adalah salah satu basis data *open-source* paling populer di dunia yang dikembangkan oleh pengembang asli MySQL. Ini berarti kode sumber dapat diunduh secara bebas dan diatur oleh lisensi yang membantu memastikan kode sumber tetap gratis dan terbuka untuk semua. MariaDB dapat di-*install* dan digunakan untuk kepentingan pribadi pada laptop maupun komputer *desktop* [19].

MariaDB adalah server basis data yang memiliki performa tinggi, *scalable* yang dilengkapi dengan berbagai fitur seperti ragam *storage engine* dan *plugin* untuk menangani berbagai kasus penggunaan. Sama seperti basis data relasional lainnya, MariaDB menyimpan data pada tabel yang terdiri dari kolom dan baris. Pengguna bisa mendefinisikan, memanipulasi, mengontrol, dan kueri data menggunakan SQL [19].

2.7 Kejahatan

Kitab Undang Undang Hukum Pidana (KUHP) tidak memberikan satupun definisi kejahatan, melainkan hanya memberikan unsur-unsur perbuatan yang dianggap sebagai suatu kejahatan. R. Soesilo membedakan pengertian kejahatan kedalam dua sudut pandang yakni sudut pandang yuridis dan sudut pandang sosiologis [20]. Dari sudut pandang yuridis, kejahatan adalah suatu perbuatan/tingkah laku yang bertentangan dengan kejahatan undang-undang. Sedangkan dari sudut pandang sosiologis, kejahatan adalah perbuatan atau tingkah laku yang selain merugikan si penderita, juga sangat merugikan masyarakat yaitu berupa hilangnya keseimbangan, ketentraman dan ketertiban.

Untuk menyebut suatu perbuatan sebagai sebuah kejahatan ada tujuh unsur pokok yang saling berkaitan yang harus dipenuhi yakni [21]:

1. Adanya perbuatan yang menimbulkan kerugian.

2. Kerugian yang tersebut telah diatur dalam Kitab Undang Undang Hukum Pidana. Contoh: misalnya orang dilarang mencuri, dimana larangannya yang menimbulkan kerugian tersebut telah diatur didalam Pasal 362 KUHP (asas legalitas).
3. Harus ada perbuatan (*criminal act*).
4. Harus ada maksud jahat (*criminal intent-mens rea*).
5. Ada peleburan antara maksud jahat dan perbuatan jahat.
6. Harus ada pembaruan antara kerugian yang telah diatur didalam KUHP tentang perbuatan
7. Harus ada sanksi pidana yang mengancam perbuatan tersebut.

2.8 Penipuan

Menurut Kamus Besar Bahasa Indonesia disebutkan bahwa tipu berarti kecoh, daya cara, perbuatan atau perkataan yang tidak jujur (bohong, palsu, dan sebagainya), dengan maksud untuk menyesatkan, mengakali, atau mencari untung. Dengan demikian maka berarti bahwa yang terlibat dalam penipuan adalah dua pihak yaitu orang menipu disebut dengan penipu dan orang yang tertipu. Jadi penipuan dapat diartikan sebagai suatu perbuatan atau membuat, perkataan seseorang yang tidak jujur atau bohong dengan maksud untuk menyesatkan atau mengakali orang lain untuk kepentingan dirinya atau kelompok [22].

Unsur-unsur suatu perbuatan sehingga dapat dikatakan sebagai penipuan dan pelakunya dapat dipidana dirumuskan kedalam pasal 378 KUHP sebagai berikut: “Barangsiapa dengan maksud untuk menguntungkan diri sendiri atau orang lain dengan melawan hukum, dengan memakai nama palsu atau martabat palsu, dengan tipu muslihat atau pun dengan rangkaian kebohongan menggerakan orang lain untuk menyerahkan sesuatu benda kepadanya, atau supaya memberi hutang maupun menghapuskan piutang, diancam karena penipuan dengan pidana penjara paling lama 4 (empat) tahun”.

Berdasarkan pengertian-pengertian diatas, tindak penipuan bisa diartikan sebagai tipu muslihat atau serangkaian perkataan bohong sehingga seseorang merasa terpedaya karena perkataan yang seakan-akan benar. Biasanya seseorang yang melakukan

penipuan, adalah menerangkan sesuatu yang seolah-olah betul atau terjadi, tetapi sesungguhnya perkataannya itu adalah tidak sesuai dengan kenyataannya, karena tujuannya hanya untuk meyakinkan orang yang menjadi sasaran agar dilakukan keinginannya [22].

Sebagai cara penipuan dalam Pasal 378 KUHP, menurut M. Sudrajat Bassar menyebutkan:

1. Menggunakan nama palsu. Dalam hal ini, penipu menggunakan nama milik orang lain dan bukan dirinya sendiri atau menggunakan nama yang tidak diketahui pemiliknya atau tidak ada pemiliknya.
2. Menggunakan kedudukan palsu. Dalam hal ini, penipu menggunakan suatu kedudukan yang mana menciptakan atau memberikan hak-hak yang sebenarnya tidak dimiliki oleh si penipu.
3. Menggunakan tipu muslihat dan/atau susunan belit dusta untuk menimbulkan kepercayaan tentang sesuatu yang sesungguhnya tidak benar.

2.9 Kenyamanan Bertelepon

Gangguan keamanan seperti SMS penipuan, SMS spam, virus, telepon penipuan merupakan contoh kecil dari banyaknya gangguan keamanan bertelepon di Indonesia. Tercatat sebanyak 91 persen dari 100 responden mengatakan bahwa dirinya pernah mengalami gangguan keamanan [23]. Adapun cara-cara yang bisa dilakukan untuk melindungi diri dari gangguan keamanan yaitu [24]:

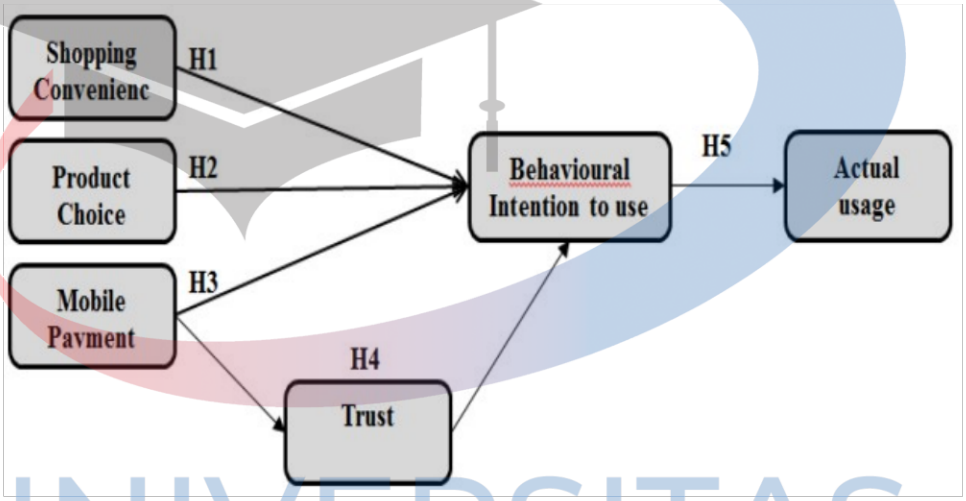
1. Berhati-hati apabila di telepon oleh nomor yang dicurigai
2. Menggunakan aplikasi pihak ketiga dalam membantu mengelola panggilan masuk
3. Menyadari akan pentingnya keamanan privasi untuk melawan para penipu

2.10 Kenyamanan Bertransaksi Online

Transaksi *online* yang dilakukan melalui situs web maupun aplikasi *mobile* masih menyisakan banyak kekhawatiran akan keamanan dan privasi seperti informasi alamat rumah, riwayat pembelian pribadi, riwayat penelusuran, transaksi yang diproses oleh

pihak ketiga (Paypal atau kartu kredit), dan sebagainya [25]. Adapun kebiasaan bertransaksi *online* dipengaruhi oleh beberapa faktor yaitu [26]:

1. Kemudahan membeli kapan dan dimanapun
2. Pilihan produk yang bervariasi, luas, dan ketersediaannya
3. Pembayaran Elektronik
4. Kepercayaan
5. Niat Perilaku dalam memutuskan faktor yang paling mempengaruhi kebiasaan bertransaksi antara poin 1 - 3



Gambar 2.12 Faktor pengaruh kebiasaan bertransaksi *online*