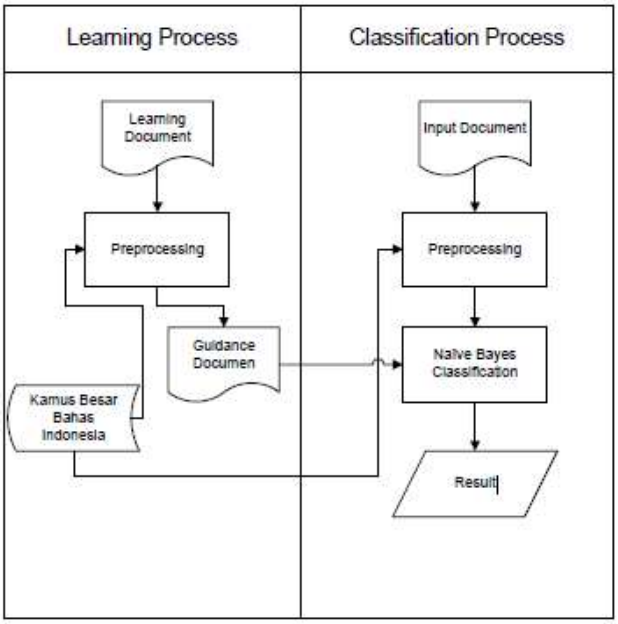
BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Ni Made Ari Lestari melakukan penelitian tentang penggunaan *naïve bayes* untuk mengklasifikasikan tipe personalitas seseorang dari sebuah dokumen yang menjelaskan tentang personalitas mereka. Bahasa yang digunakan dalam dokumen adalah bahasa Indonesia. Terdapat 2 jenis proses yang dilakukan dalam klasifikasi, yakni *Learning process* dan *Classification process*. Pada Gambar 2.1 adalah gambaran singkat penelitian yang dilakukan.



Gambar 2.1 Diagram singkat tentang penelitian

Sumber : Lestari (2013)

Dalam melakukan penelitiannya dilakukan beberapa metodologi yakni *preprocessing text, Classification* dengan *Naïve Bayes, Personality Types, Couple Compability by Type Personality.* Pada preprocessing text dijelaskan bagaiamana memproses teks bahasa Indonesia sebelum dilakukan pengklasifikasian dilakukan *case folding, tokenizing, filtering, stemming.*

Case folding adalah merubah huruf kapital pada dokumen menjadi huruf kecil dan menghilangkan komponen selain huruf "a" sampai "z". Pada tahap *tokenizing* dokumen dijadikan komponen lebih kecil, dari kalimat menjadi kata perkata. Selanjutnya pada *filtering* dihilangkan kata kata yang termasuk di dalam stopword (kata-kata yang tidak dipertimbangkan dalam pengklasifikasian). Kemudian dilakukan *stemming*. Setelah didapatkan kata kata dasar dari dokumen proses selanjutnya adalah dilakukan pengklasifikasian menggunakan *naïve bayes*. Dalam penelitiannya dilakukan pengklasifikasian terhadap tipe personalitas yang didapat

dari studi yang dilakukannya. Hasil dari eksperimen pengklasifikasian tipe personalitas telah berhasil mendapatkan tipe personalitas dan memberikan rekomendasi teman yang tepat berdasarkan tipe personalitas dengan menggunakan *text mining* dengan metode *Naïve bayes* untuk klasifikasi personalitas.

2.2 Pengertian Sistem Informasi

Sistem adalah suatu proses yang saling berkaitan yang memiliki 3 aspek yakni input, proses dan output. Sebuah sistem dikatakan baik ketika sistem tersebut bisa memberikan output yang diharapakan dari input yang direncanakan. Sistem informasi terdiri dari gabungan teknologi, fasilitas, orang-orang, media, pengendalian dan prosedur yang bertujuan untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberikan notifikasi kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dan menyediakan dasar informasi untuk pengambilan keputusan yang cerdik (Jogiyanto, 1999).

Sistem informasi merupakan teknologi yang bisa membantu kerja manusia dalam menjalankan proses bisnis yang ada pada perusahaan. Sistem informasi adalah kumpulan dari komponen-komponen yang saling berinteraksi untuk menghasilkan sebuah informasi (Kroenke, 2007). Dengan adanya sistem informasi, beberapa komponen dalam perusahaan bisa saling membagi informasi, data dan kebutuhan dari proses bisnis secara cepat dan terintegrasi. Dengan begitu antara satu proses bisnis dengan proses bisnis lain bisa dengan cepat untuk mendapatkan informasi.

2.3 KOMINFO (Kementrian Komunikasi dan Informasi)

Kementerian Komunikasi dan Informatika (sebelumnya bernama "Departemen Penerangan" (1945-1999), "Kementerian Negara Komunikasi dan Informasi" (2001-2005), dan Departemen Komunikasi dan Informatika (2005-2009), disingkat Depkominfo) adalah Departemen/kementerian dalam Pemerintah Indonesia yang membidangi urusan komunikasi dan informatika. Kementerian Kominfo dipimpin oleh seorang Menteri Komunikasi dan Informatika (Menkominfo) yang sejak tanggal 27 Oktober 2014 dijabat oleh Rudiantara.

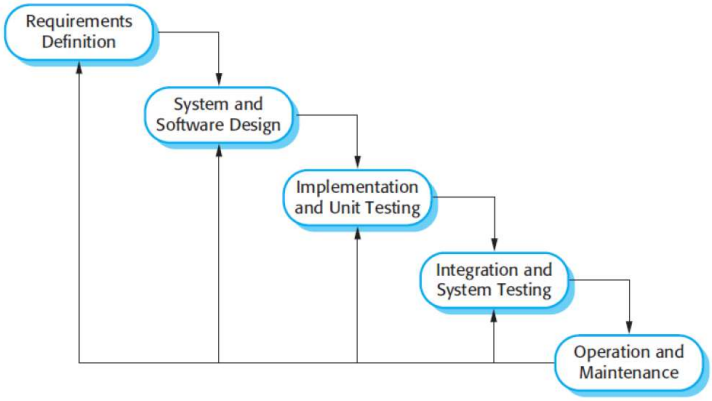
2.4 KIM (Kelompok Informasi Masyarakat)

KIM sebagai komunitas masyarakat informasi yang tumbuh dan berkembang di lingkungan masyarakat Indonesia yang diharapkan dapat berperan menjadi fasilitator untuk menjembatani kesenjangan komunikasi dan informasi yang terjadi antara pemerintah dengan masyarakat (*top down*) atau sebaliknya antara masyarakat dan pemerintah (*bottom up*). KIM sebagai agen informasi, berperan aktif berbagi informasi yang perlu diketahui oleh masyarakat, sehingga

masyarakat dapat melakukan langkah antisipatif yang bermanfataan untuk menopang aktivitas mereka.

2.5 Metode Pengembangan Model *Waterfall*

Pengembangan model *waterfall* memiliki keruntutan dari satu fase ke fase lainnya, dengan demikian model ini dikenal dengan model *waterfall*. Pada model *waterfall* perancangan dilakukan dengan terencana terlebih dahulu karena digunakan sebagai dasar pengembangan kedepannya (Sommerville, 2011).



Gambar 2.2 *Waterfall model*

Sumber: Sommerville (2011)

Pada Gambar 2.2 dapat dilihat metode pengembangan model *waterfall*. Tahapan utama dari model *waterfall* secara langsung menerapkan aktifitas dasar dalam pengembangan:

1. *Requirement analysis and definition* (Analisis kebutuhan dan definisi kebutuhan). Layanan, batasan, dan tujuan dibangun dengan berdikusi dengan pengguna sistem. Lalu dijelaskan lebih rinci dan dijadikan sebagai spesifikasi sistem.
2. *System and Software design* (Sistem dan desain perangkat lunak). Pada desain sistem meliputi pendefinisian kebutuhan perangkat keras ataupun perangkat lunak untuk arsitektur sistem secara keseluruhan. Desain perangkat lunak meliputi identifikasi dan definisi fundamental dari sebuah abtraksi *software system* dan hubungan-hubungannya.
3. *Implementation and unit testing* (Implementasi dan pengujian unit). Pada tahap ini program direalisasikan dan diuji setiap unitnya untuk memastikan program yang dibuat sesuai dengan kebutuhan yang telah didefiniskan.
4. *Integration and System Testing.* Dari unit-unit yang diitegrasikan, digabungkan diuji, kemudian setelah program sudah lengkap secara kesuluruhan sistem, program diujikan pada pengguna sistem.

5. *Operation and Maintenance.* Pada tahap ini sistem disintal dan digunakan. *Maintenance* meliputi perawatan *software*, membetulkan kesalahan-kesalahan atau *bug,* dan kekurangan-kekurangan pada program yang sebelumnya tidak terbahas pada tahap-tahap sebelumnya. Hal ini menyebabkan perulangan pada tahap sebelumnya untuk melakukan perubahan. Dalam penilitian ini tidak dilakukan tahap *operation and maintenance*.

2.6 Proses Bisnis

Proses bisnis merupakan sekumpulan aktivitas berelasi yang membutuhkan suatu *input* untuk menghasilkan *output* misal dalam bentuk laporan atau peramalan yang memiliki nilai bagi pelanggan (Monk, 2013). Proses bisnis juga memiliki definisi sebagai kumpulan aktivitas yang dirancang untuk menghasilkan *output* yang spesifik untuk pelanggan atau pasar tertentu (Sparx System, 2004). Proses bisnis memiliki komponen yaitu : (1) tujuan yang jelas, (2) adanya suatu masukan, (3) adanya suatu keluaran, (4) menggunakan sumber daya. (5) mempunyai kegiatan yang dibagi dalam beberapa tahap, (6) mempengaruhi lebih dari satu unit dalam organisasi dan (7) dapat menghasilkan nilai bagi konsumen.

2.7 *Bussines Processs Model Notation*

Business Process Modelling Notation (BPMN) adalah bentuk pemodelan untuk menentukan bagaiamana proses yang terjadi dalam alur bisnis dan siapa saja aktor yang terlibat dalam sebuah alur bisnis tersebut. BPMN digunakan untuk menyediakan notasi-notasi yang bisa dipahami oleh pengguna bisnis, dari analis bisnis sebagai perancangan proses, orang yang mengimplementasikan teknologi, dan orang bisnis sebagai pengelola dan pengontrol proses tersebut (Silver, 2012).

Terdapat 4 kategori elemen dalam BPMN elemen sebagai penunjang dalam pembuatan BPMN (Object Management Group, 2011). Kategori-kategori elemen tersebut adalah :

2.7.1 *Flow Object*

Terdapat 3 pembagian terhadap *flow object* yakni *event, activity,* dan *gateway*. Berikut penjelasannya :

1. *Event*
   Event dapat didefinisikan sebagai suatu hal yang terjadi dan berdampak dalam proses bisnis. E*vent* bisa berasal dari internal ataupun eksternal dari suatu proses. *Event* dibagi menjadi tiga yaitu *start event, intermediate event,* dan *end event*. Pada setiap proses harus memiliki sebuah *start event* untuk sebagai awal dari proses bisnis. Pada Tabel 2.1 merupakan penjelasan dari tipe *event*.

Tabel 2.1 Tipe *event*

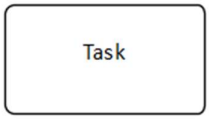
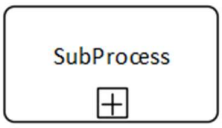
No	Tipe	Deskripsi	Simbol
1	*Start*	Proses dimulai.	
2	*Intermediate*	Terjadi diantara awal dan akhir proses. Berpengaruh pada alur dari proses, tapi tidak akan memulai atau menghentikan suatu proses.	
3	*End*	Akhir dari suatu proses.	

Sumber : Object Management Group (2011)

2. *Activity*

   Activity dapat didefinisikan sebagai tugas yang dilakukan dalam proses bisnis. *Activity* digambarkan dengan kotak yang memiliki ujung bulat dan memiliki nama yang menjelaskan aktivitas yang dilakukan. Terdapat dua macam *activity* yaitu *task* dan *sub process*. Penjelasan dari tipe *activity* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Tipe *activity*

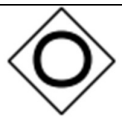
No	Tipe	Deskripsi	Simbol
1	*Task*	Aktivitas yang dilakukan pada alur proses.	
2	*Sub Process*	Aktivitas majemuk yang dimasukkan dalam proses. Aktivitas majemuk tersebut dapat dijelaskan dengan lebih rinci.	

Sumber : Object Management Group (2011)

3. *Gateway*

   Gateway memiliki peran dalam mengontrol alur dari sebuah proses bisnis. *Gateway* digambarkan dengan bentuk belah ketupat yang didalamnya terdapat simbol yang memiliki fungsi fungsi tertentu. Terdapat 4 jenis *gateway* yang dapat dilihat pada Tabel 2.3.

Tabel 2.3 Tipe *gateway*

No	Tipe	Deskripsi	Simbol
1	*Exclusive*	*divergence* : untuk membuat jalur alternatif dalam sebuah proses, tetapi hanya salah satu yang dipilih. *convergence* : untuk mengabungkan jalur alternatif.	
2	*Parallel*	Proses yang dijalankan secara bersamaan.	
3	*Inclusive*	Sebuah proses yang dipecah menjadi beberapa jalur.	
4	*Complex*	Alur kompleks pada sebuah proses bisnis.	

Sumber : Object Management Group (2011)

2.7.2 Connections

Connections dapat didefinisikan sebagai elemen penguhubung *flow objects*. *Connections* memiliki 3 tipe yang masing masing memiliki bentuk yang berbeda dan fungsi yang berbeda.Pada Tabel 2.4 merupakan penjelasan dari tipe *connections*.

Tabel 2.4 Tipe c*onnections*

No	Tipe	Deskripsi	Simbol
1	*Sequence Flow*	Menunjukan urutan aktivitas yang dilakukan pada suatu proses.	
2	*Association*	Menunjukan hubungan antara teks, *database*, artifak lain, dan *flow object* pada suatu proses.	
3	*Message Flow*	Menunjukan alur pesan antara dua partisipan yang dapat mengirim dan menerima pesan.	

Sumber : Object Management Group (2011)

2.7.3 Swimlanes

Swimlanes adalah wadah grafis yang mengelompokan suatu set aktivitas dengan aktivitas lainnya. Terdapat 2 tipe swimlane yakni *pool* dan *lane*. Pada Tabel 2.5 merupakan penjelasan tipe *swimlanes*.

Tabel 2.5 Tipe s*wimlanes*

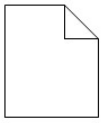
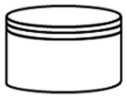
No	Tipe	Deskripsi	Simbol
1	*Pool*	Wadah yang berisikan satu proses dan *sequence flow* yang menjadi penguhubung antar aktivitas.	
2	*Lane*	Mempresentasikan tanggungjawab aktivitas pada suatu proses.	

Sumber : Object Management Group (2011)

2.7.4 Artifacts

Artifacts adalah objek luar dari sebuah proses. *Artifact* dapat berupa data atau catatan yang dapat digunakan untuk mengelola tugas atau proses. Terdapat 4 tipe *artifacts* yang akan dijelaskan pada Tabel 2.6.

Tabel 2.6 Tipe *artifacts*

No	Tipe	Deskripsi	Simbol
1	*Data Object*	Informasi yang mengalir pada sebuah proses misal dokumen bisnis, *email*, surat dan lain-lain.	
2	*Data Store*	Suatu tempat yang dapat digunakan dalam sebuah proses untuk membaca atau menulis data.	
3	*Annotation*	Menunjukan informasi kepada pembaca berupa diagram BPMN.	
4	*Group*	Digunakan untuk mengelompokkan elemen secara informal.	

Sumber : Object Management Group (2011)

2.8 *Levels Of Requirement*

Menurut (IBM Corporation Rational Software, 2011) ada dua tingkatan sudut pandang dalam melakukan analisis kebutuhan yakni *business perspective* dan *user perspective*.

Gambar 2.3 Levels of requirements

Sumber : Diadaptasi dari IBM Corporation Rational Software (2011)

2.8.1 *Business Perspective*

Business Perspective adalah sudut pandang yang melihat kebutuhan pengembangan aplikasi dari sisi fungsi dan proses bisnis untuk memenuhi tujuan bisnis yang ingin dicapai. Tujuannya adalah untuk memberikan pemahaman terhadap pemangku kepentingan bagaimana alur proses bisnis yang ada pada perusahaan tersebut yang bertujuan untuk mendapatkan gambaran secara jelas bagaimana proses bisnis yang sudah berjalan dan bagian mana dari proses bisnis tersebut yang akan ditingkatkan kinerjanya dengan menggunakan aplikasi tersebut sehingga mendapatkan kebutuhan yang mendukung tujuan bisnis yang ingin dicapai. *Business Perspective* dapat disebut juga sebagai domain masalah.

2.8.1.1 *Business Process (As-Is vs To-Be)*

Business Process as-is menjelaskan secara keseluruhan proses bisnis yang berjalan saat ini di sebuah perusahaan atau organisasi dengan menggunakan pernyataan dan gambar. Tujuan dari analisis *Business Process as-is* untuk memperjelas dan memahami seperti apa kondisi saat ini dari proses bisnis yang ada dan bagaimana alur bisnis tersebut berjalan pada perusahaan atau organisasi (Bridging the Gap, 2017).

Proses analisis yang dilakukan untuk mengetahui permasalahan dan kelemahan yang ada pada proses bisnis tersebut. Permasalahan tersebut dipetakan dalam tabel analisis permasalahan, sehingga bisa diketahui bagaimana solusi dan perbaikan untuk proses bisnis tersebut. Pada Tabel 2.7 merupakan cara dalam melakukan analisis permasalahan.

Tabel 2.7 Analisis permasalahan

Masalah	[Mendeskripsikan masalah]
Mempengaruhi	[Pemangku kepentingan yang terpengaruh oleh masalah]
Dampak	[Dampak dari masalah]
Solusi	[Solusi beserta manfaatnya]

Sumber : Bittner (2002)

Business Process to-be menjelaskan bagaimana peningkatan kinerja proses bisnis kedepannya kepada pemangku kepentingan perusahaan atau organisasi. Usulan yang diajukan berdasarkan apa yang sudah dilakukan pada analisis *Business Process as-is* untuk memenuhi seluruh permasalahan yang ada pada analisis *Business Process as-is.* Tujuan dari analisis *Business Process to-be* adalah untuk memberikan penjelasan dan gambaran kepada pemangku kepentingan perusahaan atau organisasi bagaimana usulan-usulan tersebut disatukan dengan proses bisnis yang ada, apakah sudah memenuhi keinginan dari pemangku kepentingan yang terlibat dalam proses bisnis tersebut (Bridging the Gap, 2017).

2.8.1.2 *Kebutuhan Pemangku Kepentingan*

Menurut Sommerville dan Sawyer menjelaskan bahwa pemangku kepentingan atau pemangku kepentingan adalah seseorang yang mendapatkan manfaat secara langsung maupun tidak langsung dari sistem yang dikembangkan (Somerville, 1997). Identifikasi kebutuhan dari pemangku kepentingan dilakukan untuk mengetahui bagaimana hubungan antara aktor dengan masalah-masalah yang ada sebelum diterapkannya aplikasi atau sistem. Dengan mengetahui siapa saja pemangku kepentingan yang akan menggunakan sistem tersebut, pengembang aplikasi bisa menentukan fungsi dan fitur apa saja yang ada dalam aplikasi dengan menyesuaikan karakteristik dari pemangku kepentingan tersebut, dimana pemangku kepentingan bertindak sebagai *user* atau pengguna dari sistem. Beberapa langkah dilakukan dalam analisis kebutuhan pemangku kepentingan adalah :

1. Tipe Pemangku Kepentingan

   Tipe Pemangku kepentingan adalah kumpulan pemangku kepentingan yang memiliki karakteristik dan hubungan yang sama dengan sistem atau aplikasi yang dikembangkan. Pada Tabel 2.8 merupakan cara dalam melakukan analisis tipe pemangku kepentingan

Tabel 2.8 Analisis tipe pemangku kepentingan

Tipe Pemangku Kepentingan	Deskripsi	Pemangku Kepentingan
[Tipe pemangku kepentingan]	[Menjelaskan tipe pemangku kepentingan]	[Menyebutkan pemangku kepentingan yang termasuk dalam tipe pemangku kepentingan]

Sumber: Diadaptasi dari Bittner (2003)

2. Peran dan Perwakilan Pemangku Kepentingan

Peran pemangku kepentingan merupakan kumpulan pemangku kepentingan yang memiliki peran dan tanggung jawab yang sama terhadap pengembangan aplikasi. Hal tersebut bertujuan untuk mengetahui peran dan tanggung jawab masing-masing pemangku kepentingan sehingga menghindari kesalahan komunikasi pada saat proyek dilaksanakan.

Pemangku kepentingan sangat penting dalam proyek karena sistem yang dibuat digunakan secara langsung oleh pemangku kepentingan. Tabel 2.9 merupakan cara dalam melakukan analisis peran dan perwakilan pemangku kepentingan.

Tabel 2.9 Peran dan perwakilan pemangku pepentingan

Peran Pemangku Kepentingan	Perwakilan Pemangku Kepentingan	Deskripsi
[Peran pemangku kepentingan]	[Perwakilan pemangku kepentingan]	[Menjelaskan peran pemangku kepentingan]

Sumber: Diadaptasi Bittner (2003)

3. Peran Pengguna

Peran pengguna merupakan peran yang dilakukan oleh pengguna saat berinteraksi dengan fungsi fungsi yang ada di dalam sistem. Peran pengguna juga dapat menggambarkan sistem lain yang melakukan interaksi dengan sistem.

4. Kebutuhan Pemangku Kepentingan dan Pengguna

Kebutuhan pemangku kepentingan menggambarkan masalah dalam proses bisnis atau pribadi dari pemangku kepentingan yang perlu dilakukan analisis bagaimana solusi yang dibutuhkan untuk mendukung tujuan bisnis dan menyelesaikan masalah yang ada. Identifikasi kebutuhan pemangku kepentingan dapat memberikan sebuah pandangan terhadap masalah utama dan solusi yang dapat menyelesaikan masalah tersebut (Bittner, 2003).

2.8.2 *User Perspective*

User Perspective adalah pandangan dari pengguna untuk membuat solusi untuk mengatasi masalah yang ada di *business perspective*.

2.8.2.1 *Fitur*

Fitur adalah suatu kemampuan sistem yang bernilai bagi pengguna serta memenuhi kebutuhan pemangku kepentingan dan pengguna. Suatu fitur dapat memiliki beberapa kebutuhan fungsional (Bittner, 2002). Fitur Tabel 2.10 merupakan cara dalam menjelaskan fitur perangkat lunak.

Tabel 2.10 Fitur sistem

Kode Fitur	Fitur	Deskripsi
[Kode suatu fitur, sebagai tiap fitur]	[Fitur sistem]	[Deskripsi dari fitur]

Sumber : Bittner (2002)

2.8.2.2 *Kebutuhan Fungsional*

Kebutuhan fungsional dari sistem menggambarkan apa dan bagimana sistem yang harus dilakukan oleh sebuah sistem. Kebutuhan-kebutuhan tersebut bergantung pada tipe aplikasi yang akan dikembangkan, apa yang diinginkan oleh pengguna dari aplikasi yang dikembangkan, dan bagaimana pendekatan dari pemangku kepentingan ketika menentukan kebutuhan aplikasi. Ketika menjelaskan kebutuhan dari pemangku kepentingan atau pengguna, Kebutuhan fungsional adalah salah satu cara untuk menjelaskan fungsi yang diberikan kepada pengguna yang bisa dipahami (Sommerville, 2011).

2.8.2.3 *Kebutuhan Non-Fungsional*

Kebutuhan non-fungsional adalah kebutuhan yang tidak langsung berkaitan dengan layanan yang diberikan oleh sistem kepada pengguna. Kebutuhan ini berkaitan dengan keandalan sistem, waktu proses dari sistem, dan keamanan sistem. Kebutuhan non fungsional juga menjelaskan bagaimana sebuah sistem dikembangkan sesuai dengan keadaan atau kondisi eksternal dari sistem seperti kebutuhan sistem untuk merespon perangkat *input output*, bagaimana sistem bisa menampilkan data yang representatif dan sesuai dengan keinginan pengguna, dan bagaimana sistem bisa menyesuaikan dengan sistem lain (Sommerville, 2011).

2.9 *Framework* Laravel

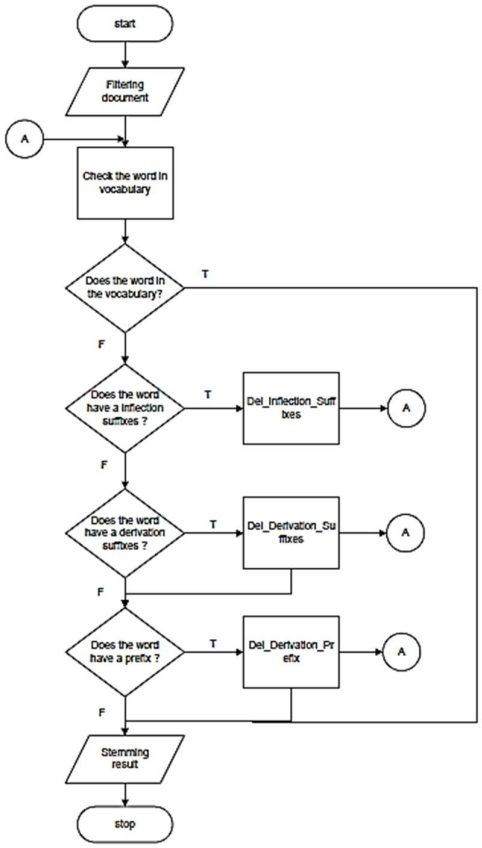
Framework laravel adalah salah satu *framework* php yang tersedia saat ini. Laravel terbentuk dari beberapa komponen seperti carbon, symphony dan masih banyak lagi. Sampai saat ini laravel masih terus dikembangkan sampai versi 5.4. Di komunitas terdapat banyak *library* yang tersedia, sehingga proses pengembangan perangkat lunak dapat lebih cepat dengan pemanfaatan *reuseable asset* dari para *developer* lain.

2.10 *Text Mining*

Text mining adalah analisi teks yang digunakan untuk menggali informasi atau pengetahuan dari dokumen berupa teks, bisa disebut dengan istilah *Knowledge Discovery in Text*. Tujuan *Text mining* adalah menemukan sebuah pola atau tren yang dapat bermanfaat dari sekumpulan dokumen. Fungsi utama dari *text mining* meliputi *information extraction* (IE), kategorisasi teks, *summarization*, *text clustering*, *information monitor,* dan *question and answer* (Mustafa, 2009). Dibandingkan dengan *data mining*, *text mining* memerlukan tahapan yang lebih banyak. Hal ini disebabkan karena *text mining* memilki karakteristik yang lebih kompleks dari pada data biasa yakni data yang tidak terstruktu, sehingga perlu dilakukan pengubahan data menjadi lebih terstruktur.

2.11 *Stemming* Bahasa Indonesia

Stemming adalah penghilangan imbuhan pada kata sehingga keluarannya berupa kata kata dasar. Pada Gambar 2.4 dapat dilihat proses *stemming* pada bahasa Indonesia.



Gambar 2.4 Stemming algorithm

Sumber : Lestari (2013)

Metode *stemming* pada bahasa Indonesia dimulai dengan memasukkan suatu kata. Kemudian kata tersebut dilanjutkan dengan proses pengecekan kata dasar. Jika kata tersebut sudah terdapat pada kata dasar maka kata tersebut menjadi keluaranya, ketika kata tersebut tidak terdapat, maka akan diproses dengna pengecekan lebih dalam. Terdapat beberapa proses yang akan dilakukan yakni menghapus *inflection suffixes,* menghapus *derivation suffixes,* menghapus *prefix derivation.* Pada *prefix* terdapat perkecualian-perkecualian terhadap kata kata yang berubah bentuk ketika mendapat imbuhan sehingga membutuhkan proses tambahan untuk mengecek kata tersebut, misal me+sesal=menyesal. Jadi ketika input kata menyesal output dari *stemming* adalah sesal.

2.11.1 Library *Stemming* Bahasa Indonesia (Sastrawati)

Sastrawi adalah library PHP sederhana yang memungkinkan untuk mengurangi kata-kata infleksi di Bahasa Indonesia (Bahasa Indonesia) ke bentuk basis mereka. Algoritma yang digunakan pada library ini adalah hak intelektual masing-masing pemiliknya yakni algoritma Nazief dan Adriani, Asian J. dengan *Effective Techniques for Indonesian Text Retrieval*, Arifin, A.Z., I.P.A.K. Mahendra dan H.T. Ciptaningtyas dengan *Enhanced Confix Stripping Stemmer and Ants Algorithm for Classifying News Document in Indonesian Language*, A. D. Tahitoe, D. Purwitasari dengan Implementasi Modifikasi *Enhanced Confix Stripping Stemmer* Untuk Bahasa Indonesia dengan Metode *Corpus Based Stemming*. Kemudian untuk meningkatkan kualitas kode, algoritma tersebut diterapkan ke dalam *Object Oriented Design*.

2.12 *Spamming*

Spamming adalah kegiatan mengirim *email* dalam jumlah berlebih dan bisa disebut pesan sampah dan aktifitas tersebut tidak ilegal, namun menjadi hal yang kurang etis (Wilen,1996). *Spam* juga merupakan informasi yang tidak berguna(sampah) yang dikirimkan melalui berbagai pihak ke alamat *email* (Hakim,2013). Pada umumnya *spam* berisi iklan komersial, surat berantai, pornografi, ajakan dan berbagai isu lainnya, tetapi informasi tersebut tidak memiliki hubungan apapun dengan kepentingan dan minat pengguna. Dalam perkembangannya *spam* tidak hanya terjadi dalam *email* saja, melainkan juga terjadi di media lain, misal media sosial, *blog*, iklan. Dalam penelitian ini *spam* digunakan sebagai kategori informasi yang kurang sesuai dengan apa yang diharapkan.

2.13 Library PHP Classifier

PHP Classifier adalah klasifikasi perpustakaan teks dengan focus pada penggunaan kembali, *customizability* dan kinerja. Classifiers dapat digunakan untuk berbagai tujuan, tetapi sangat berguna dalam mendeteksi *spam*.

Beberapa fitur yang terdapat pada PHP Classifier

1. Complement Naive Bayes Classifier, digunakan untuk melakukan pengkasifikasian menggunakan *naïve bayes*.
2. Multiple types of model caching, digunakan untuk untuk menyimpan file cache yang menyimpan perhitungan *naïve bayes* dari data train yang pernah dilakukan.

2.14 Unified Model Language (UML)

UML adalah bahasa standar yang digunakan dan memvisualisasikan artefak dari proses analisis dan desain berorientasi objek. UML hanya merupakan bahasa pemodelan dan tidak menggambarkan bagaimana melakukan analisis dan desain berorientasi objek. Dalam pemodelan suatu sistem UML menyediakan standar pada notasi dan diagram. Pemodelan UML tidak didominasi oleh narasi, pemodelan dapat dilakukan secara visual. Pemodelan secara visual dapat membatu menangkap struktur dan perilaku dari objek, mempermudah penggambaran interaksi antar elemen dalam sistem, dan mempertahankan konsistensi antara desain dan implementasi dalam pemrograman (Hermawan, 2015). Menurut Yasin (2013), ada dua tujuan penggunaan UML yaitu:

1. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi objek.
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

Secara konseptual, terdapat tiga unsur utama pada UML yaitu :

1. *Things*, pada UML terdapat empat jenis hal yaitu *Structural*, *Behavioral*, *Grouping*, dan *Annotation*.
2. *Relationship*, merupakan hubungan antar elemen pada sebuah pemodelan UML. Terdapat empat jenis relasi pada UML yaitu dependensi, asosiasi, generalisasi, dan realisasi.
3. Diagram, merupakan presentasi grafis dari serangkaian elemen yang sering diterjemahkan sebagai graf terhubung simpul (*things*) dan jalur (*relationship*). Terdapat 13 jenis diagram yang termasuk pada UML, diantaranya adalah *Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Use case diagram, Sequence diagram, Communication Diagram, State Diagram, Activity diagram, Deployment Diagram, Package Diagram, Timing Diagram, dan Interaction Overview Diagram.*

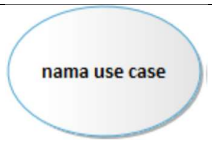
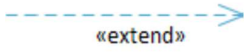
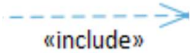
2.14.1 *Use case diagram*

Pengertian *Use case diagram* digunakan untuk menggambarkan interaksi antara pengguna dengan sistem (Booch, et al., 2007). U*se case* menjelaskan apa saja yang dilakukan oleh aktor dan sistem tanpa menjelaskan bagaimana aktor dan sistem melakukan kegiatan tersebut.

Spesifikasi *use case* terdiri dari nama *use case*, deskripsi singkat (*Brief Description*), Aliran Normal (*Basic Flow*), Aliran Alternatif (*Alternate Flow*), *Special Requirement*, *Pre-Condition* dan *Post-Condition* (Hermawan, 2015). Dalam langkah langkah yang dideskripsikan harus memiliki urutan yang normal atau disebut main flow dan urutan yang dideskripsikan berjalan ketika sebuah proses berjalan dengan benar. Kemudian dilanjutkan dengan menjelaskan *alternative flow* yang mendeskripsikan sebuah perkecualian, sebuah penanganan pada kesalahan yang bisa disebut *exception*, dan bisa juga kemungkinan pengguna mengeksekusi *use case* pada skenario yang berbeda. Ketika struktur eksekusi *use case* memiliki variasi yang banyak, maka *variant* bisa didefinisikan pada *main flow.* Terdapat komponen lain untuk memperluar deskripsi dari use yakni *precondition*, *post condition*, dan pemangku kepentingan (Wazlawick,2014).

Use case adalah unit bagian dari fungsionalitas sistem yang menyediakan *classifier* dan urutan pertukaran pesan oleh subjek dan satu atau lebih aktor dalam unit sistem. Sebuah *use case* melibatkan interaksi aktor dan sistem atau subjek lainnya (Booch, et al., 2005). Simbol-simbol yang digunakan pada diagram *use case* dapat dilihat pada Tabel 2.10

Tabel 2.10 Simbol-simbol diagram *use case*

Nama	Simbol	Deskripsi
Use case		Spesifikasi tingkah laku sebuah entitas terhadap interaksi yang terjadi di luar.
Aktor / *actor*		Penggambaran peran dari pengguna di luar sistem, sistem atau sub sistem terhadap sistem. Setiap aktor berpartisipasi terhadap satu atau lebih *use case* di dalam sistem. Seorang aktor bisa berupa manusia, sistem komputer atau proses eksekusi yang lainnya.
Asosiasi / *association*		Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / extend		Tingkah laku yang hanya berjalan jika ada kondisi tertentu.
Menggunakan / *include*		Tingkah laku yang harus dipenuhi supaya event dapat terjadi dan *use case* tersebut merupakan bagian dari *use case* lainnya

Tabel 2.10 Simbol-simbol diagram *use case*

Nama	Simbol	Deskripsi
Generalisasi / *generalization*		Hubungan antar *use case* yang umum dan lebih spesifik.

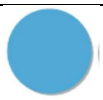
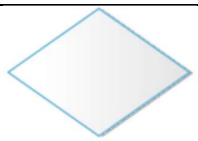
Sumber : Booch (2005)

Dengan menggunakan UML, diagram *use case* dapat diterapkan untuk menggambarkan perilaku sistem, sub sistem, atau *class* sehingga pengguna dapat memahami bagaimana dapat memahami bagaimana menggunakan elemen tersebut dan juga pengembang dapat mengimplementasikan elemen tersebut.

2.14.2 Diagram Aktivitas

Diagram aktivitas adalah salah satu dari limas diagram di UML untuk memodelkan aspek dinamis dari sistem. Diagram aktivitas pada umumnya merupakan sebuah *flowchart* yang menunjukkan aliran kontrol dari aktivitas ke aktivitas lainnya. Diagram aktivitas menunjukkan *concurrency* serta percabangan (Booch, 2005). Tabel 2.11 merupakan keterangan dari simbol-simbol yang digunakan pada diagram aktivitas.

Tabel 2.11 Simbol-simbol diagram aktivitas

Nama	Simbol	Deskripsi
Status awal		Status awal aktivitas dari sistem
Aktivitas		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Branch /cabang		Sebuah cabang mungkin memiliki satu arus masuk dan dua atau lebih arus keluar dan digambarkan dalam simbol *diamond*. Pada setiap aliran keluar, ditempatkan ekspresi *Boolean*, yang dievaluasi saat memasuki cabang. Pada kondisi tersebut cabang berperan sebagai *decision*. Ketika dua aliran kontrol kembali bergabung bersama-sama, simbol *diamond* dapat digunakan dengan dua arus masuk dan satu arus keluar.

Tabel 2.11 Simbol-simbol diagram aktivitas (lanjutan)

Nama	Simbol	Deskripsi
		Pada kondisi ini cabang berperan sebagai *merge*.
Fork-join		*Fork* merupakan pemisahan alur kontrol tunggal menjadi dua atau lebih alur kontrol bersamaan. Pada *fork*, mungkin memiliki satu transisi masuk dan dua atau lebih transisi keluar, masing-masing merepresentasikan aliran kontrol yang independen. *Join* merupakan sinkronisasi dua atau lebih alur kontrol yang bersamaan. Pada *join* mungkin memiliki dua atau lebih transisi masuk dan satu transisi keluar.
Status akhir		Status akhir yang dilakukan dari sistem
Swimlane		Bertujuan untuk partisi aktivitas pada diagram aktivitas dalam kelompok-kelompok, setiap kelompok mewakili organisasi bisnis yang bertanggung jawab.

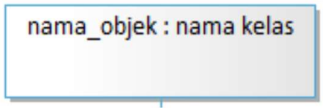
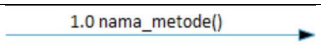
Sumber : Booch (2005)

2.14.3 Sequence diagram

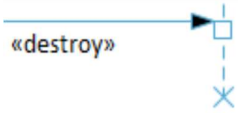
Sequence diagram atau diagram sekuen menggambarkan interaksi antara objek di dalam dan di sekitar sistem (termasuk pengguna, tampilan, dll.) Dalam bentuk pesan atau pesan yang digambarkan terhadap waktu. Diagram sekuen yang biasa menggambarkan skenario atau rangkaian langkah yang dilakukan dalam menanggapi suatu peristiwa untuk menghasilkan keluaran tertentu. Pesan yang dijelaskan dalam diagram sekuen akan dipetakan ke fungsi / metode *class* pada tahap perancangan selanjutnya. Untuk objek yang memiliki sifat khusus, standar UML mendefinisikan simbol khusus untuk objek *boundary, controller* dan *persistent entity* (Yasin, 2012).

Diagram sekuen merupakan diagram interaksi yang menekankan pada saat permintaan pesan yang menunjukkan serangkaian peranan dan pesan yang dikirim dan diterima oleh bagian yang ada pada peranan tersebut. Diagram sekuen digunakan untuk menggambarkan *dynamic view* dari sistem (Booch, 2005). Simbol-simbol yang digunakan pada diagram *sequence* ditunjukkan pada Tabel 2.12

Tabel 2.12 Simbol-simbol diagram *sequence*

Nama	Simbol	Deskripsi
Lifeline		Menunjukkan keberadaan objek selama periode waktu.
Objek		Menyatakan objek yang berinteraksi menggunakan pesan atau *message.*
Activation Bar		Menunjukkan periode waktu selama sebuah objek melakukan kegiatan, baik secara langsung maupun melalui prosedur perintah.
Pesan tipe *call* (*synchronous*)		Menyatakan suatu objek memanggil fungsi yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki fungsi, karena ini memanggil sebuah fungsi maka fungsi yang dipanggil harus ada pada *class diagram* sesuai dengan *class* objek yang berinteraksi.
Pesan tipe *send* (*asynchronous*)		Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya

Tabel 2.12 Simbol-simbol diagram *sequence* (lanjutan)

Nama	Simbol	Deskripsi
Pesan tipe *return*		Menyatakan bahwa suatu objek yang telah menjalankan suatu fungsi menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe *destroy*		Objek dapat diakhiri selama interaksi yang berlangsung. *Lifeline* objek berakhir apabila menerima pesan *destroy.* Bila ada pesan tipe *create* maka sebaiknya ada pesan tipe *destroy.*

Sumber : Booch (2005)

2.14.4 Class Diagram

Class Diagram merupakan suatu spesifikasi, apabila spesifikasi tersebut diinstasiasi akan menghasil objek yang merupakan inti dari pengembangan berorientasi objek. *Class diagram* menggambarkan (atribut/properti) dari perangkat lunak, sekaligus berisi layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain (Yasin, 2012). *Class diagram* dapat digunakan sebagai diagram model domain dan diagram kelas implementasi (Fakhroutdinov, 2009)

Suatu atribut dan fungsi suatu *class* dapat memiliki salah satu dari sifat berikut ini :

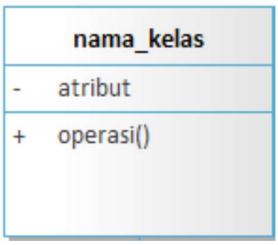
a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
c. *Public*, dapat dipanggil oleh siapa saja (Yasin, 2012).

Class diagram digunakan untuk memodelkan tampilan statis perancangan dari sebuah sistem. *Class diagram* tidak hanya penting untuk memvisualisasikan, menentukan, dan mendokumentasikan model struktural, tapi juga penting untuk menyusun sistem yang dapat dijalankan melalui teknik *forward* maupun *reverse*. *Class diagram* secara umum memuat tiga hal, yaitu *class*, *interfaces*, dan hubungan dependensi, generalisasi, serta asosiasi (Booch, 2005).

a. *Class*, adalah deskripsi dari sekumpulan objek yang memiliki atribut, operasi, hubungan, dan semantik yang sama. Sebuah *class* mengimplementasikan satu atau lebih *interfaces.*
b. *Interfaces*, dalam UML *interfaces* digunakan untuk memodelkan lapisan dalam suatu sistem yang merupakan kumpulan dari operasi yang digunakan untuk menentukan sebuah layanan dari *class* atau komponen. Dengan mendeklarasikan antarmuka maka dapat menyatakan perilaku dari independen abstraksi dari sebuah implementasi abstraksi tersebut.
c. Dependensi, merupakan hubungan yang menyatakan bahwa sebuah hal (*class*) menggunakan informasi dan layanan dari hal lain (*class* lain), tetapi belum tentu sebaliknya. Biasanya menggunakan dependensi antar *class* untuk menunjukkan bahwa satu *class* menggunakan operasi dari *class* lain atau menggunakan variabel yang ada pada *class* lain.
d. Generalisasi, adalah hubungan antara sebuah hal yang umum (disebut *superclass* atau *parent*) dan jenis hal yang lebih spesifik (disebut *subclass* atau *child*). Objek pada *child class* mungkin digunakan untuk variabel atau tipe parameter oleh *parent class*, tapi tidak sebaliknya. Hal ini berarti *child class* disubstitusikan untuk deklarasi *parent class*.
e. Asosiasi, merupakan hubungan struktural yang menentukan bahwa objek dari satu *class* yang terhubung ke objek *class* lain. (Booch, 2005)

Pada Tabel 2.13 menjelaskan tentang simbol-simbol yang digunakan dalam pembuatan *class diagram* pada umumnya.

Tabel 2.13 Simbol-simbol diagram *class*

Nama	Simbol	Deskripsi
Class		*Class* pada suatu struktur sistem.
Asosiasi / *association*		Relasi antar *class* dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Ketergantungan / *dependency*		Ketergantungan antar *class*.
Agregasi / *aggregation*		Relasi antar *class* dengan makna semua-bagian (*whole-part*).

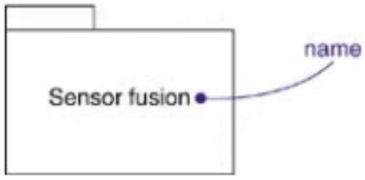
Tabel 2.13 Simbol-simbol diagram *class*

Nama	Simbol	Deskripsi
Asosiasi berarah / *directed association*		Relasi antar *class* dengan makna *class* yang satu digunakan oleh *class* yang lain.
Generalisasi		Relasi antar *class* dengan makna *class* yang satu digunakan oleh *class* yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

Sumber : Booch (2005)

2.14.5 *Package*

Dalam UML, potongan-potongan yang mengatur suatu model disebut *package* atau paket. Sebuah paket adalah tujuan umum mekanisme untuk mengorganisir elemen dalam kelompok-kelompok. Paket membantu untuk mengatur elemen dalam model sehingga lebih mudah untuk dipahami (Booch, 2005). UML menyediakan representasi grafis dari paket, seperti yang terlihat dalam Gambar 2.5.



Gambar 2.5 Representasi *package*

Sumber : Booch (2005)

Setiap paket harus memiliki nama yang dapat membedakannya dengan paket lain yang berupa teks. Sebuah paket mungkin memiliki unsur-unsur elemen tersendiri, termasuk *class*, *interfaces*, komponen, *node*, kolaborasi, *use case,* diagram, dan bahkan paket lain.

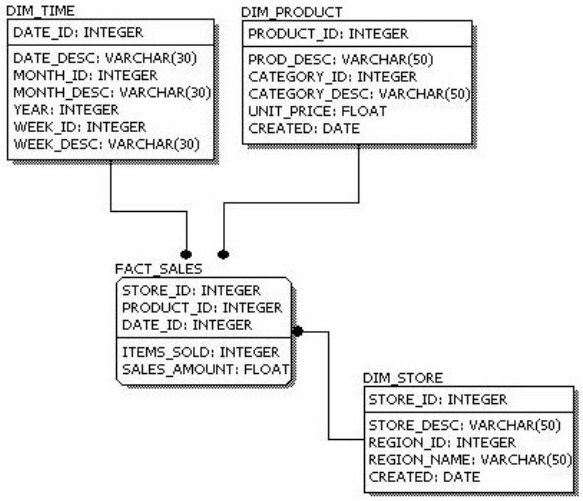
2.15 Physical Data Model

Physical data model merepresentasikan bagaimana model akan dibangun di *database*. Sebuah *physical data model* menampilkan struktur tabel yang terdiri dari nama kolum, tipe data kolum, *contraints* kolum, *primary key, foreign key,* dan relasi antara tabel (1keydata, 2017). Terdapat beberapa tahap dalam pembuatan *physical data model,* yakni

1. Merubah entitas menjadi tabel.
2. Merubah relasi menjadi *foreign key.*

3. Merubah atribut menjadi kolum.
4. Memodifikasi *physical data model* sesuai dengan kebutuhan.

Pada Gambar 2.6 dapat dilihat contoh *physical data model.*



Gambar 2.6 *Physical data model*

Sumber : 1keydata(2017)

2.16 Pengujian White-*Box*

Pengujian *white-box testing* merupakan sebuah filosofi perancangan *test case* yang menggunakan struktur kontrol yang dijelaskan sebagai bagian dari perancangan peringkat komponen untuk menghasilkan *test-case* (Pressman, 2010).

Teknik yang digunakan dalam pengujian white-box menurut Tom McCabe adalah uji *basis path*. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logika dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. Test case yg didapat digunakan untuk mengerjakan basis set yg menjamin pengerjaan setiap perintah minimal satu kali selama uji coba (Chaerani, 2005).

Cyclomatic complexity adalah metrik PL yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam kontek metode uji coba *basis path*, nilai yang dihitung untuk *cyclomatic complexity* menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali (Chaerani, 2005).

Dengan menggunakan pengujian *white-box* dapat memperoleh *test case* yang (1) menjamin bahwa semua jalur indipenden di dalam modul telah dieksekusi

sedikitnya satu kali. (2) melaksanakan semua keputusan logis pada sisi benar dan yang salah. (3) melaksanakan semua *loop* pada batas mereka dan dalam batas-batas operasional mereka, dan (4) melakukan strukur data internal untuk memastikan kesalahannya.

2.17 Pengujian *Black-Box*

Pegujian *black-box* berfokus pada persyaratan fungsional dari sebuah perangkat lunak. Pada teknik pengujian ini akan ditentukan berbagai kondisi inputan yang akan dimasukan pada program, dimana *input* tersebut akan memenuhi semua tugas (menguji) dari tiap fungsional program. Pengujian ini bermaksud untuk mengetahui apabila terdapat kesalahan dalam fungsi program, antarmuka, kesalahan pada data atau akses basis data, kesalahan performa serta kesalahan pada inisialisasi dan terminasi program (Pressman, 2010).

Black box testing menggunakan *validation testing* yang digunakan untuk menguji kebutuhan fungsional sedangkan untuk menguji kebutuhan non-fungsional akan menggunakan *compability testing.*

1. *Validation Testing*

   Pengujian validasi dilakukan untuk mengetahui apakah sistem yang dibanung berjalan sesuai dengan kebutuhan yang telah didefinisikan. Pengujian validasi termasuk kedalam *black-box testing*, karena pengujian dilakukan tanpa menunjukan proses berjalannya sebuah fungsi sebenarnya tetapi pengujian validasi melihat hasil yang dikeluarkan apakah sesuai dengan kebutuhan atau tidak dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan (Indriati, 2010).

2. *User Acceptance Testing*

   User Acceptance Testing (UAT) merupakan pengujian yang menangani kebutuhan pengguna, *requirement,* dan *business process.* UAT dilakukan untuk mengetahui apakah suatu sistem telah memenuhi kriteria untuk diterima oleh pengguna atau *customer* (Hambling, 2013).

   UAT biasanya dilakukan sebelum produk dihidupkan dan dilakukan setelah pengujian sistem. UAT dilakukan oleh *user* yang terlibat dengan sistem yang dibuat. Terdapat bebera poin yang perlu diperhatikan dalam membuat UAT (softwaretestinghelp, 2007)

   1. UAT bukan tentang halaman, *fields,* tombol. Asumsi tersebut dilakukan bahkan sebelum UAT dimulai. Semua hal mendasar itu seharusnya telah diuji dan bekerja dengan baik.
   2. UAT adalah tentang entitas yang merupakan elemen utama dalam bisnis

3. UAT juga merupakan bentuk pengujian pada inti sistem yang berarti ada kemungkinan bagus untuk mengidentifikasi beberapa bug pada fase ini juga
4. UAT diklasifikasikan sebagai pengujian Alpha dan Beta, namun klasifikasi tersebut tidak begitu penting dalam konteks proyek pengembangan perangkat lunak biasa di industri berbasis layanan.
5. Sebagian besar waktu dalam proyek pengembangan perangkat lunak biasa, UAT dilakukan di lingkungan *quality assurance* jika tidak ada lingkungan pementasan atau UAT.

3. Pengujian kompabilitas

Pengujian kompabilitas dilakukan untuk mengetahui apakah sistem yang dibangun dapat berjalan pada lingkungan yang berbeda. Beberapa tipe pengujian *compability* antara lain (Guru99, 2016a) *hardware, operating systems, software, network, browser, devices, mobile, version.*

Pengujian kompabilitas yang dilakukan pada penelitian ini adalah pengujian kompabilitas *browser.* Dari pengujian kompabilitas akan didapatkan *browser* apa saja yang mendukung sistem yang dibuat berjalan dengan baik.

Pada pengujian kompabilitas ini dilakukan menggunakan bantuan *tool* yakni SortSite. SortSite akan melakukan pengujian kompabilitas secara otomatis pada sistem yang dibuat dengan memasukkan url dari alamat wes sistem yang dibangun. Selain itu, SortSite mampu menguji perangkat lunak berbasis *website* pada beberapa *checkpoints* diantaranya *accessibility, broken links, compatibility, search engine optimation, privacy, web standart,* dan *usability* (PowerMapper, 2016).

2.18 Metrik kinerja untuk Modeling prediktif

Dalam kasus klasifikasi, pengukuran yang dapat digunakan adalah penggunaan *coincidence matrix* (Olson, 2008). Pada Tabel 2.14 dapat dilihat penggambaran *simple coincidence matrix*.

True Positive Rate (TPR)= $\frac{TP}{TP+F}$ (2.1)

True Negative Rate (TNR) = $\frac{TN}{TN+FP}$ (2.2)

Accuracy =$\frac{TP+}{TP+TN+FP+F}$ (2.3)

Precision=$\frac{TP}{TP+FP}$ (2.4)

Recall=$\frac{TP}{TP+FN}$ (2.5)

F-measure = $\frac{2}{\frac{1}{\text{Presition}}+\frac{1}{\text{Recall}}}$ (2.6)

Pada Tabel 2.15 adalah contoh matrix klasifikasi untuk 2 kelas kasus klasifikasi. TPR dihitung dengan menggunakan rumus 9.1 dengan hasil 88%, sementara TNR dihitung dengan menggunakan rumus 9.2 dengan hasil 62%. Keseluruhan akurasi dari contoh klasifikasi yang dihitung dengan menggunakan rumus 9.3 adalah 82%.

		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

Tabel 2.14 *Simple coincidence matrix*

Sumber : David L.Olson, Dursun Delen (2008)

		Actual Classification of Classes in the Dataset		
		Class 1	*Class 2*	
Model Classification	*Class 1*	674	91	
	Class 2	89	146	
	Sum	30	30	
	Probability	0.76	0.24	
	Accuracy	0.88	0.62	0.82

Tabel 2.15 *Contoh coincidence matrix* untuk 2 kelas klasifikasi

Sumber : David L.Olson, Dursun Delen (2008)