BAB 2

TINJAUAN PUSTAKA

2.1 *Sentiment Analysis*

Sentiment Analysis digunakan untuk mengetahui emosi (positif, negatif, dan netral) seseorang yang dilepaskan melalui tulisan. Biasanya tulisan tersebut didapatkan melalui internet seperti media sosial. Selanjutnya, dapat di analisis sentimen apa yang dimiliki oleh si penulis dari tulisannya. Pada kali ini *sentiment analysis* bisa dilakukan dengan otomatis menggunakan komputer. Maka, dapat berguna jika ingin mengetahui emosi apa saja yang diutarakan melalui tulisan oleh seseorang kepada perusahaan, pemerintahan, dan lain sebagainya. Target dari sentiment analysis adalah untuk menemukan opini, mengidentifikasi sentimen yang mereka ungkapkan, dan kemudian mengklasifikasikan polaritasnya [8].

2.2 Kata Hinaan

Hinaan merupakan dasar dari kata hina, yang memiliki banyak makna seperti rendah kedudukannya, keji, tercela, dan tidak baik. Yang dimaksud kata dasar adalah kata yang menjadi dasar pembentukan kata berikutnya [9]. Seperti, kata *hina* menjadi kata dasar dari kata *hinaan*. Kata *hina* dapat digunakan kepada orang lain dan juga ke diri sendiri. Maka dari itu, kata *hina* jarang diucapkan karena memiliki makna yang negatif.

Arti dari Kata *hinaan* itu sendiri menurut Kamus Besar Bahasa Indonesia (KBBI) adalah cercaan, nistaan. Biasanya kata *hinaan* digunakan untuk mengatakan sesuatu kepada orang lain, tidak sama dengan kata *hina* yang bisa juga digunakan pada diri sendiri. Sebagai contoh, kalimat “Wah kamu baik sekali, ya? Tapi, sayang pas ada maunya aja” merupakan sindiran keras kepada seseorang. Kemudian, orang yang mendengar sindiran itu merasa terganggu. Oleh karena itu, kata *hinaan* dapat mengganggu, menyinggung, dan membuat marah orang lain.

Kata *hinaan* akan terlihat atau terdeteksi dengan jelas, jika diterapkan dalam suatu kalimat, atau berupa kelompok kata. Sebuah kalimat akan memperjelas maksud dari ucapan atau tulisan yang mengandung kata *hinaan*.

2.3 Algoritma *K-Nearest Neighbor* (KNN)

K-Nearest Neighbor dalah metode non-parametrik yang digunakan untuk klasifikasi dan regresi [10]. Klasifikasi yang dilakukan terhadap sekumpulan data berdasarkan pembelajaran yang sudah terklasifikasi sebelumnya. Tujuan dari algoritma ini adalah mengklasifikasikan objek baru berdasarkan jarak suatu objek yang akan diklasifikasikan terhadap data contoh [11]. KNN termasuk ke dalam *supervised learning*, karena *dataset* akan diberikan label agar dapat mengidentifikasi secara eksplisit dan melakukan prediksi atau klasifikasi yang sesuai. Untuk membuat prediksi pada data baru, algoritma ini akan mencari data points paling dekat di dalam *training dataset*. Oleh sebab itu, pengklasifikasian algoritma k-NN berdasarkan kategori mayoritas tetangga terdekat ke-K.

Perhitungan jarak antara data uji dan data latih akan menggunakan rumus *Euclidean distance*, agar dapat menentukan jarak tetangga terdekat ke-K.Untuk menghitung jarak antara dua titik bisa digunakan jarak Euclidean sebagai berikut:

$$d(x,y) = || x - y ||^2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (2.1)$$

Keterangan :

$d(x,y)$: Jarak *Euclidean* (*Euclidean Distance*).

x_i: data x ke- i

y_i: data y ke- i

i: 1,2,3,...n

Setelah hasil dari perhitungan *Euclidean distances* didapatkan, maka akan dilakukan pengurutan. Pengurutan ini diurut secara *ascending*, atau dari kecil ke

besar. Supaya dapat menentukan tetangga terdekat berdasarkan jarak minimum ke-K.

Algoritma KNN dalam hal ini disarankan untuk memberikan nilai ganjil terhadap K, karena akan mempengaruhi hasil. Jika menggunakan nilai genap maka tidak bisa menemukan hasil mayoritas kelasnya. Sedangkan, nilai ganjil sudah pasti memiliki hasil yang berbeda dan bisa menemukan mayoritas kategori terdekatnya. Contoh, kita beri nilai 3 untuk K dan hasil yang didapat adalah "2 untuk kelas 0" dan "1 untuk kelas 1". Maka dari itu, kelas 0 menjadi hasil dari klasifikasi karena memiliki 2 *vote*.

Adapun langkah-langkah dalam klasifikasi pada algoritma KNN:

1. Tentukan parameter K = jumlah tetangga terdekat.
2. Hitung jarak antara data baru dengan semua data *training*.
3. Urutkan jarak tersebut dan tetapkan tetangga terdekat berdasarkan jarak minimum ke-K.
4. Periksa kelas dari tetangga terdekat.
5. Gunakan mayoritas sederhana dari kelas tetangga terdekat sebagai nilai prediksi data baru.

2.4 *Local Mean Based K-Nearest Neighbor* (LMKNN)

Metode *local mean-based k-nearest neighbor* telah diusulkan oleh Mitani dan Hamamoto [12]. Karena keefektifan dan desain pengklasifikasian LMKNN yang mudah, inti dari idenya telah berhasil diterapkan ke banyak metode peningkatan yang lain. Aturan dari metode ini adalah pengklasifikasian non parametrik yang sederhana, efektif dan kuat. Hasil dari LMKNN menunjukan bahwa dapat meningkatkan kinerja klasifikasi serta bisa mengurangi pengaruh *outlier* yang ada, terutama untuk jumlah data yang kecil.

Adapun langkah-langkah atau alur kerja LMKNN adalah sebagai berikut:

1. Menentukan Nilai K.
2. Menghitung jarak data uji keseluruh data latih dengan menggunakan *euclidean distance*.

3. Urutkan jarak antar data dari yang terkecil ke yang terbesar sebanyak k dari setiap kelas.
4. Melakukan perhitungan *local mean vector* dari setiap kelas dengan persamaan:

$$m_{wj}^{k} = \frac{i}{j} \sum_{i=1}^{k} y_{i,j}^{NN} \tag{2.2}$$

5. Tentukan kelas data uji dengan cara menghitung jarak terdekat ke *local mean vector* dari setiap kelas data.

$$w_c = argmin_{w\ j} d(x, m_{wj}^{k}), j = 1,2,\ldots,M \tag{2.3}$$

Nilai K pada LMKNN sangat berbeda jauh dari K-NN, pada LMKNN nilai K merupakan jumlah tetangga terdekat dari setiap kelas data, sedangkan pada K-NN nilai K merupakan jumlah tetangga terdekat dari seluruh data. LMKNN sama dengan 1-NN jika nilai K bernilai 1 [6].

2.5 Metode *Preprocessing*

Text preprocessing merupakan pengolahan data korpus atau mentah yang akan diproses untuk meningkatkan performa dari sistem *Information Retrieval* (IR). Teknik *preprocessing* akan menghilangkan gangguan dari data teks, kemudian mengidentifikasi kata dasar untuk kata yang sebenarnya dan mengurangi ukuran data teks [13]. Tujuan di balik *text preprocessing* adalah untuk mewakili setiap dokumen sebagai vektor fitur yaitu untuk membagi teks menjadi kata-kata individual [14]. Data masukan yang berupa dokumen atau teks yang tidak terstruktur, maka dari itu *text preprocessing* penting dilakukan. Tahapan *preprocessing* yang akan dilakukan dalam penelitian kali ini meliputi *case folding*, *normalization*, *noise removal*, *stopword*, *convert negation*, *tokenization*.

2.6.1 *Case Folding*

Proses *case folding* menyeragamkan semua teks ke dalam satu bentuk standar baik menjadi huruf kecil (*lowercase*) atau huruf besar (*uppercase*). Biasanya akan diseragamkan semua teks menjadi huruf kecil, begitupun dengan penelitian ini. Berikut adalah contoh proses *case folding*.

Contoh proses *case folding* dapat dilihat dibawah ini:

Teks awal:

> **Mobil** lejen .buriq. bikin sakit mata. apalagi skin, :) yg
 permainan tida tertarik </br>.

Setelah *case folding*:

> **mobil** lejen .buriq. bikin sakit mata. apalagi skin, :) yg
 permainan tida tertarik </br>.

2.6.2 *Normalization*

Normalization adalah suatu proses mengubah kata yang tidak baku ke dalam kata baku atau formal. Contohnya seperti “abis”, “adlh”, “akikah”, “akyu”, “asek”, “asoy”, dan lain-lain. Untuk dapat melakukan proses pengklasifikasi kata hinaan, kata tidak baku tersebut harus diubah ke dalam kata baku. Pada penelitian ini akan menggunakan kamus data sebagai acuan kata formal bahasa Indonesia, dapat dilihat pada tabel 2.1.

Tabel 2.1 Contoh daftar kata formal

Kata Tidak Baku	Kata Formal
7an	tujuan
@	di
ababil	abg labil
abis	habis
acc	accord
ad	ada

adlah	adalah
adlh	adalah
adoh	aduh
afaik	as far as i know
aha	tertawa
ahaha	haha
aing	saya
aj	saja
aja	saja
ajep	dunia gemerlap
ajj	saja
ak	saya
aka	dikenal juga sebagai
akika	aku
akko	aku
akkoh	aku
akku	aku
apasih	apa sih
ape	apa
apes	sial
aplot	unggah
aps	apa
apva	apa
aq	saya
aqu	aku
aquwh	aku
asap	sesegera mungkin
asbun	asal bunyi
aseek	asyik
aseekk	asyik
asek	asyik
asekk	asyik
aseknya	asyiknya

asem	asam
asoy	asyik
aspal	asli tetapi palsu
astrojim	astagfirullahaladzim
astul	asal tulis
ath	kalau begitu
ato	atau
atuh	kalau begitu
au	ah tidak mau tahu
ava	avatar
awak	saya
aws	awas
ay	sayang
ayang	sayang
ayank	sayang
ayok	ayo
b4	sebelum
bacot	banyak bicara
bakalan	akan
bales	balas
bandes	bantuan desa
bangdes	pembangunan desa
bangedh	banget
bangkotan	tua
banpol	bantuan polisi
banpres	bantuan presiden
banpur	bantuan tempur
bansarkas	bantuan sarana kesehatan
basbang	basi
bazis	badan amal, zakat, infak, dan sedekah
bcanda	bercanda
bcoz	karena

bdg	bandung
beb	sayang
begajulan	nakal
bejibun	banyak
beliin	belikan
belom	belum
bencong	banci
bener	benar
bentar	sebentar
ber2	berdua
ber3	bertiga
berdikari	berdiri di atas kaki sendiri
beresin	membereskan
bet	banget
bete	bosan
beti	beda tipis
beud	banget
beut	banget
bg	abang
bgd	banget
bgmn	bagaimana
bgs	bagus
bgt	banget
bhubu	tidur
bijimane	bagaimana
bimbuluh	bimbingan dan penyuluhan
bintal	bimbingan mental
bisi	kalau-kalau
bkl	akan
bkn	bukan
bknnya	bukannya
bl	beli
blegug	bodoh

blg	bilang
blh	boleh
blm	belum
bln	bulan
bls	balas
blum	belum
bnchi	benci
bnci	benci
bngung	bingung
bnran	yang benar
bnyk	banyak
bodor	lucu
bohay	badan aduhai
bokap	ayah
bokep	video porno
boker	buang air besar
bokin	pacar
bokis	bohong
bole	boleh
boljug	boleh juga
bolot	bodoh
bonek	bocah nekat
bonyok	ayah ibu
boyeh	boleh
bpk	bapak
br	baru
brb	segera kembali
brg	bareng
brngkt	berangkat
bro	saudara laki-laki
brp	berapa
bru	baru
brur	saudara laki-laki

bs	bisa
bsa	bisa
bsen	bosan
bsk	besok
bt	bosan sekali
bt	buat
btw	ngomong-ngomong
bu_bu	tidur
buaya	tidak setia
bubarin	bubarkan
bubbu	tidur
buber	buka bersama
bubu	tidur
bujubune	luar biasa
bumil	ibu hamil
buser	buru sergap
bw	bawa
bwhn	bawahan
bwt	buat
byar	bayar
byk	banyak
byr	bayar
byrin	bayarkan
c8	chat
cabal	sabar
cabut	pergi
cadas	keren
caem	cakep
calo	makelar
cama	cama sama-sama
can	belum
cangcut	celana dalam
capcus	pergi

cape	capek
caper	cari perhatian
caur	jelek
ce	cewek

Selanjutnya, akan menghapus huruf yang duplikat pada sebuah kata. Karena, hal tersebut merupakan sebuah kata yang tidak formal juga. Maka dari itu, proses penghapusan huruf yang duplikat penting dilakukan. Tidak hanya kata saja, akan tetapi suatu emotikon juga akan diubah ke bentuk teks. Melakukan perubahan pada emotikon dengan mengecek bentuk emotikon pada suatu teks. Pada tabel 2.2 memberikan gambaran atau bentuk-bentuk emotikon jika diubah ke dalam teks. Merubah emotikon berguna jika ingin mendapatkan suatu makna dari teks yang lebih jelas, karena *emoticon* tidak dihapus. Ringkasnya, *normalization* penting dilakukan agar dapat membersihkan kata dari tidak formal ke formal dan karakter ke teks.

Tabel 2.2 Daftar *emoticon*

Emoticon	**Konversi**
>:] :-) :) :o) :] :3 :c) :> =] 8) =) :} :^)	Senang
>:D :-D :D 8- D 8D x-D xD XD XD =-D =D =-3 =3	Tertawa
>:[:-(:(:-c :c :-< :< :-[:[:{ > .><. <> .< :’(	Sedih
D :< D : D 8 D ; D = D X v.v D-':	Horror
>:P :-P :P X-P x-p xp XP :- p :p =p :-Þ :Þ :-b :b	Tongue
>:o>:O :-O :O °o° °O° :O o_O o.O 8-0	Shock
>:\ >:/ :-/ :-. :/ :\ =/ =\ :S	Kesal
:| :-|	Ekspresi Datar

Contoh proses *normalization* dapat dilihat dibawah ini:

Case folding:

mobil lejen .buriq. bikin sakit mata. apalagi skin, **:) yg**   permainan tida tertarik </br>.

Setelah *normalization*:

mobil lejen .buriq. bikin sakit mata. apalagi skin, **senang yang**   permainan tida tertarik </br>.

2.6.3 *Noise Removal*

Noise removal adalah proses membersihkan dokumen dari komponen-komponen yang tidak memiliki hubungan dengan informasi yang ada pada dokumen, seperti karakter spesial, tag html, *link*, *script*, angka, tanda baca, dsb. Hal-hal tersebut sering ditemukan pada kolom komentar di sosial media.

Contoh proses *noise removal* dapat dilihat dibawah ini:

Normalization:

mobil lejen **.**buriq**.** bikin sakit mata**.** apalagi skin**,** senang yang ** ** permainan tidak tertarik **</br>.**

Setelah *noise removal*:

mobil lejen buriq bikin sakit mata apalagi skin senang yang permainan tidak tertarik

2.6.4 *Stopword*

Stopword merupakan suatu proses yang akan menghilangkan kata *stopword*. Kata *stopword* yang dimaksud merupakan kata-kata tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. Contohnya seperti “di”, “oleh”, “sebuah”, “karena”, dan lain sebagainya.

Contoh proses *stopword* dapat dilihat dibawah ini:

Noise removal:

mobil lejen buriq bikin sakit mata apalagi skin senang yang permainan tidak tertarik

Setelah *stopword*:

mobil lejen buriq sakit skin senang permainan tida tertarik

2.6.5 *Convert Negation*

Convert negation merupakan suatu proses yang bertujuan untuk menggabungkan kata negasi dengan kata setelahnya, misal “ga mau” menjadi “gamau”. Kata yang termasuk kata negasi memiliki jumlah sebanyak 9 kata, dapat dilihat pada tabel 2.3. Proses ini dilakukan karena dapat menentukan hasil deteksi teks.

Tabel 2.3 Daftar kata negasi

No	Kata Negasi
1	ga
2	ngga
3	tidak
4	bkn
5	tida
6	tak
7	jangan
8	enggak
9	gak

Contoh proses *convert negation* dapat dilihat dibawah ini:

Stopword:

mobil lejen buriq sakit skin senang permainan **tida tertarik**

Setelah *convert negation*:

mobil lejen buriq sakit skin senang permainan **tidatertarik**

2.6.6 *Tokenization*

Tokenization bertujuan untuk membagi teks dokumen menjadi urutan token [15]. Token adalah kata-kata yang dipisah pisah dari teks aslinya tanpa mempertimbangkan adanya duplikasi. Pengecekan dilakukan dari karakter pertama sampai karakter terakhir. Apabila ditemukan karakter penggal atau delimiter seperti spasi, tanda baca, maka teks akan dipisahkan.

Contoh proses *tokenization* dapat dilihat dibawah ini:

Convert negation:

mobil lejen buriq sakit skin senang permainan tidatertarik

Setelah *tokenization*:

“mobil”, “lejen”, “buriq”, “sakit”, “skin”, “senang”, “permainan”, “tidatertarik”

2.6 *Feature Extraction*

Ekstraksi fitur merupakan proses ekstraksi untuk mengidentifikasi entitas-entitas yang dimaksud. Ekstraksi fitur akan dilakukan menggunakan data hasil *text preprocessing* dengan TF-IDF (*Term Frequency - Inverse Document Frequency*) sebagai pembobotan katanya. TF-IDF adalah statistik numerik yang menunjukkan relevansi kata kunci dengan beberapa dokumen tertentu atau dapat dikatakan, menyediakan kata kunci tersebut, yang dengannya beberapa dokumen tertentu dapat diidentifikasi atau dikategorikan [16]. Metode ini menggabungkan dua perhitungan *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) dengan cara mengalikannya.

TF menentukan bobot term pada teks berdasarkan jumlah kemunculan dalam dokumen tersebut. Semakin besar jumlah kemunculan suatu *term* (TF tinggi) dalam dokumen, semakin besar pula bobotnya atau akan memberikan nilai kesesuaian yang semakin besar. Terdapat beberapa jenis formula diantaranya TF biner (*binary* TF), TF murni (*raw* TF), TF logaritmik, dan TF normalisasi. Formula yang akan digunakan pada penelitian kali ini adalah TF normalisasi, dapat dilihat pada rumus 2.4.

$$TF = 0.5 + 0.5\left[\frac{f_{t,d}}{max\{f_{t',d:t',d\in d}\}}\right] \quad (2.4)$$

IDF merupakan suatu pembobotan kata yang mengacu pada pengurangan dominasi *term* yang sering bermunculan. Hal ini diperlukan karena *term* yang sering muncul dalam dokumen dapat dianggap sebagai *term* umum, sehingga tidak penting nilainya. Semakin sedikit jumlah dokumen yang mengandung *term* yang dimaksud, maka nilai IDF semakin besar. Formula dari IDF yang akan digunakan pada penelitian ini dapat dilihat pada rumus 2.5.

$$IDF_j = log\,\frac{D}{df_j} \quad (2.5)$$

Tahap terakhir tinggal mengalikan hasil TF dengan hasil IDF dengan menggunakan rumus sebagai berikut:

$$w_{i,j} = TF * idf_j \quad (2.6)$$

2.7 *K-Fold Cross Validation*

K-fold adalah salah satu metode *Cross Validation* yang populer dengan melipat data sebanyak K dan mengulangi (iterasi) eksperimenya sebanyak K juga. Contohnya, data yang dimiliki sebanyak 100 dan K bernilai 5, maka akan

dilakukan proses prediksi sebanyak 5 kali. Terakhir, hasil prediksi tersebut dibagi dengan K untuk mendapatkan hasil akurasi yang maksimal. Teknik ini digunakan untuk melakukan prediksi model dan memperkirakan seberapa akurat model prediktif ketika dijalankan dalam praktiknya [17]. Dengan tujuan memperoleh hasil akurasi yang maksimal dari model tersebut.

Secara keseluruhan 5 atau *10-fold cross validation* sama-sama direkomendasikan dan telah disepakati bersama [7]. Menghitung nilai akurasi dari klasifikasi dapat dilakukan dengan menggunakan rumus berikut:

$$Akurasi = \frac{Jumlah\ klasifikasi\ benar}{jumlah\ data\ uji} \tag{2.7}$$

Rumus diatas merupakan cara untuk mendapatkan akurasi dari hasil klasifikasi bagi satu *fold*. Selanjutnya tinggal menghitung rata-rata dari hasil akurasi pada setiap *fold*.

2.8 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem yang berorientasi objek [18]. UML menerapkan level abstraksi dan proses pengembangan tidak tergantung bahasa juga teknologi. Dibandingkan dengan pendekatan berorientasi logika, diagram UML memiliki keuntungan bahwa (perangkat lunak) mengenal formalisme dan banyak alat yang ada untuk menyusun dan memanipulasi spesifikasi UML [19]. UML bukan merupakan bahasa pemrograman visual, tetapi bahasa pemodelan visual.

2.9.1 *Use Case Diagram*

Diagram *use case* adalah salah satu diagram yang digunakan untuk memodelkan aspek tingkah laku sistem. Diagram ini menggambarkan hubungkan satu atau lebih *user* (*actor*) terhadap tingkah laku (*use case*) sistem. Aktor didefinisikan sebagai tipe *user* yang akan menjalankan atau menggunakan sistem,

bisa terdapat lebih dari satu *actor*. *Use case* digunakan agar mengetahui tingkah laku apa saja yang sistem lakukan terhadap *user*, jika akan menggunakan sistem.

Terdapat 3 jenis relasi untuk menghubungkan *actor* dengan *use case* yaitu:

1. *Association*

   Teknik mengidentifikasi interaksi yang dilakukan oleh aktor tertentu dengan *use case* tertentu. Hal ini digambarkan dengan garis antara aktor terhadap *use case* tersebut.

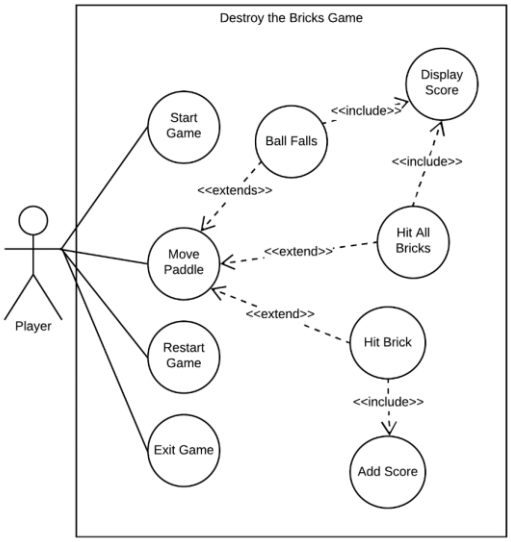
2. *Generalization*

   Mendefinisikan relasi antara dua aktor atau dua *use case* yang mana salah satunya meng-inherit dan menambahkan atau *override* sifat dari yang lainnya.

3. *Dependency*

   Dependency ini terbagi menjadi 2 macam, yaitu *include* dan juga *extend*. *Include* memiliki fungsi untuk membuat *use case* selanjutnya diharuskan untuk di eksekusi. *Extend* digunakan apabila memerlukan kondisi tertentu setelah *use case* tersebut di eksekusi.

Use case berguna untuk mengetahui fungsi yang terdapat di dalam sistem informasi dan siapa saja pengguna fungsi-fungsi tersebut. Contoh *use case* dapat dilihat pada gambar 2.1.



Gambar 2.1 Contoh *use case diagram*

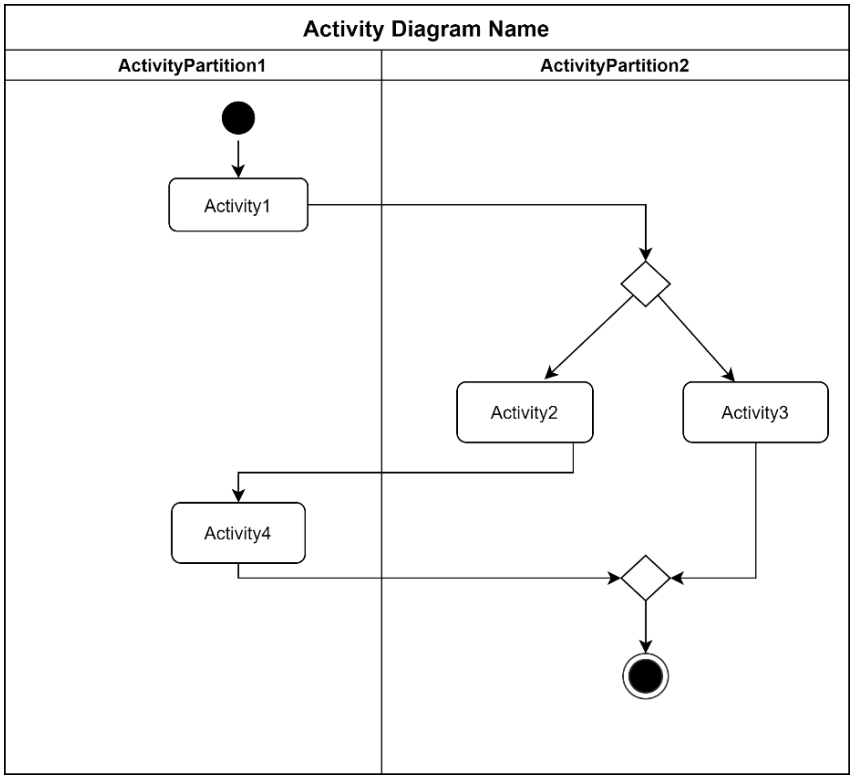
2.9.2 *Use Case Scenario*

Use case scenario merupakan penjelasan secara rinci dari setiap *use case* yang terdapat pada *use case diagram*. *Use case scenario* terbagi menjadi tiga bagian, diantaranya:

1. Identifikasi dan inisiasi
2. *Step performed*
3. Kondisi, asumsi dan pertanyaan.

2.9.3 *Activity Diagram*

Activity diagram merupakan diagram yang menggambarkan *workflow* (alur kerja) atau aktivitas yang terjadi pada suatu sistem. Aktivitas yang disebut berupa runtutan menu-menu atau proses bisnis yang terdapat di dalam sistem tersebut. *Activity diagram* berupa operasi-operasi dan aktivitas-aktivitas pada satu *use case*. Mengacu kepada *use case scenario* untuk dapat menggambarkan alur kerja yang sudah dibuat sebelumnya. Contoh *activity diagram* dapat dilihat pada gambar 2.2.



Gambar 2.2 Contoh *activity diagram*

2.9.4 *Class Diagram*

Class diagram digunakan untuk menampilkan kelas-kelas yang terdapat pada sebuah sistem. Kelas didalamnya memiliki beberapa bagian yaitu atribut dan metode, yang mana mendukung fungsionalitas sistem. Pada suatu perangkat lunak yang berbasis objek tentunya memiliki kelas-kelas yang saling terhubung satu sama lain. Maka dari itu, *class diagram* dapat menampilkan gambaran umum skema dari program, baik sederhana maupun kompleks. Contoh *class diagram* dapat dilihat pada gambar 2.3.

Elemen-elemen penting yang ada pada *class diagram* akan dijabarkan dan dijelaskan sebagai berikut:

1. Kelas

   Kelas merupakan elemen terpenting di sistem berorientasi objek. Kelas mendeskripsikan satu blok pembangunan sistem. Berikut adalah bagian dari kelas:

   a. Nama

      Nama kelas harus unik karena akan menjadi identifier di program serta tidak boleh kata kerja.

   b. Atribut

      Atribut menggambarkan sifat-sifat tertentu yang sesuai dengan objek tersebut. Kelas dapat mempunyai beberapa atribut atau tidak sama sekali.

   c. Operasi

      Operasi kelas (metode) adalah tingkah laku yang dimiliki oleh suatu objek, dan dapat digunakan oleh kelas yang lain. Operasi-operasi tersebut harus sesuai dengan kelas dimana dia ditempatkan.

   d. *Access Modifier*

      Access modifier dalam kelas digunakan untuk membatasi akses dari kelas lain yang ingin mengakses atribut atau operasi dari suatu kelas. Ada tiga *access modifier* yang digunakan yaitu *public* (+), *protected* (#) dan *private* (-).

2. Antarmuka

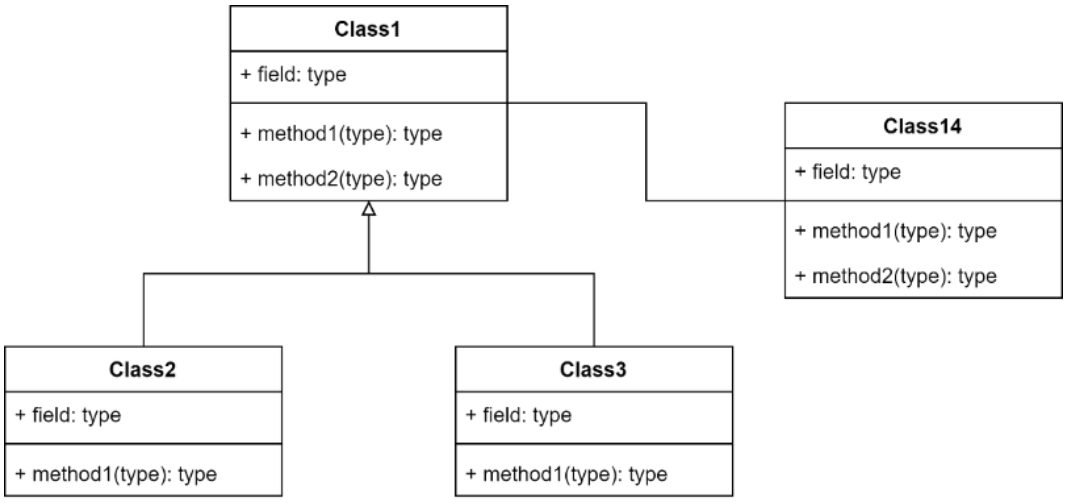
   Antarmuka (*Interface*) merupakan suatu *class* akan menangani tampilan-tampilan yang terdapat pada sistem dan mengatur tampilan untuk pengguna.

3. Kolaborasi

   Kolaborasi merupakan pendefinisian suatu interaksi dan sekelompok peran elemen-elemen lain yang bekerja sama untuk menyediakan suatu perilaku kooperatif yang lebih besar dari penjumlahan seluruh elemen. Kolaborasi memiliki struktur, perilaku dan dimensi.

4. Hubungan

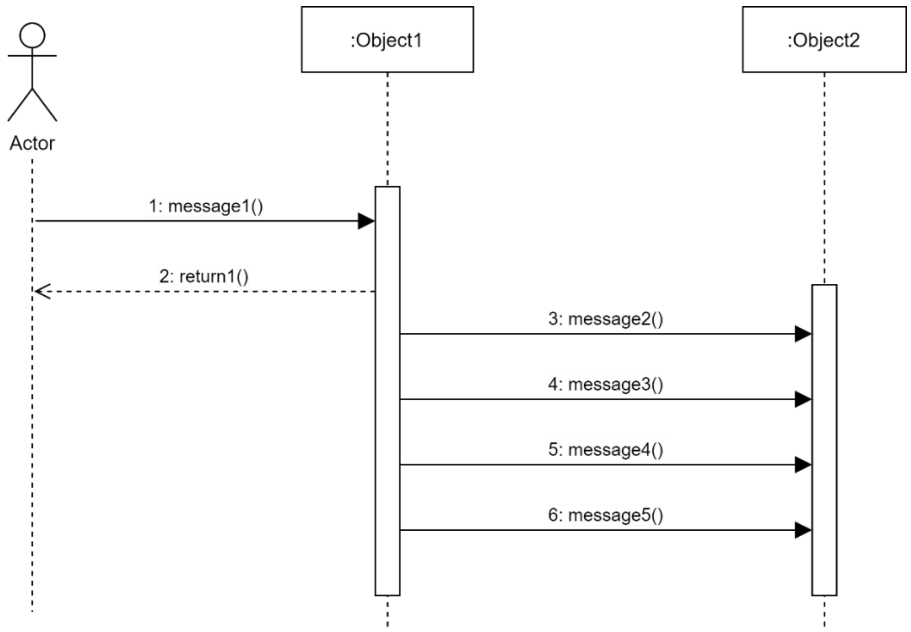
   Hubungan antar kelas diagram terdiri dari asosiasi, generalisasi, dependensi, dan agregasi.



Gambar 2.3 Contoh *class diagram*

2.9.5 *Sequence Diagram*

Sequence diagram dapat memodelkan skenario penggunaan sistem. Menjelaskan suatu tahapan dan menunjukkan rangkaian pesan yang dikirim antara objek, juga interaksi terhadap objek. Mengacu kepada *use case* agar mengetahui objek-objek yang nanti akan terlibat, satu *use case* akan menjadi satu *sequence diagram*. Contoh sequence diagram dapat dilihat pada gambar 2.4.



Gambar 2.4 Contoh *sequence diagram*

2.9 Bahasa Pemrograman

Bahasa pemrograman digunakan untuk berinteraksi dengan komputer. Komputer tidak bisa memahami bahasa manusia, sehingga diperlukan bahasa komputer dalam program komputer. Kita dapat memberikan perintah terhadap komputer melalui bahasa pemrograman ini. Bahasa pemrograman yang akan digunakan pada penelitian kali ini adalah Python.

Python merupakan bahasa pemrograman yang dapat melakukan eksekusi sejumlah instruksi multi guna secara langsung. Python mudah digunakan, tetapi ini adalah benar-benar bahasa pemrograman, menawarkan lebih banyak struktur dan dukungan untuk program besar daripada yang dapat ditawarkan oleh *shell scrip*ts atau *batch files* [20]. Python dibuat oleh programmer Belanda bernama Guido Van Rossum. Python juga dapat diterapkan dalam pembuatan *software* seperti program GUI, program CLI, *Internet of Things*, *games* dan lain-lainnya.

2.10 Youtube

Youtube merupakan sebuah aplikasi *video sharing online* baik berupa *website* maupun *mobile apps*. Youtube merupakan platform konten, sehingga jaringan sosial yang diberikan berbeda dengan yang lain dalam hal penautan

pengguna dan perilaku interaksi [21]. Tempat ini menempatkan permintaan yang signifikan pada server dan sumber daya jaringan [22]. Popularitas sosial di YouTube lebih berkorelasi dengan popularitas konten maksimum yang dicapai, bukan dengan ringkasan ukuran popularitas konten [21]. Sehingga, *video* yang dibagikan juga tidak hanya memiliki konten itu-itu saja.

Adapun fitur yang terdapat di dalam youtube tidak hanya berbagi video saja akan tetapi pengguna juga bisa melakukan *live streaming*. Pada April 2011, YouTube mengumumkan peluncuran YouTube Live dan bulan Februari 2017 diluncurkan di *mobile app*. Sekarang kita bisa melihat di Youtube banyaknya yang melakukan streaming baik itu *game*, *coding*, dan lain sebagainya.