**BAB II**

**LANDASAN TEORI**

**2.1 *E-Commerce***

*E-commerce* merupakan suatu kumpulan teknologi, aplikasi, dan proses bisnis yang menghubungkan perusahaan, konsumen, dan komunitas tertentu melalui transaksi elektronik dan perdagangan barang, pelayanan, dan informasi yang dilakukan secara elektronik (Witono & Hendrayana, 2011). Pendapat lain mendefinisikan *e-commerce* sebagai media membagikan informasi bisnis, mengelola relasi bisnis, dan mengalirkan transaksi bisnis dengan menggunakan jaringan telekomunikasi. Suatu *e-commerce* juga memungkinkan organisasi atau perusahaan untuk mempromosikan produk dan layanan mereka secara *online* dan menyediakan berbagai layanan yang pelanggan sendiri dapat lakukan (Wiradinata, 2013).

*E-commerce* merupakan bagian dari *e-business*, di mana *e-business* menggunakan teknologi internet sebagai kekuatan utamanya untuk dapat melakukan beragam aktivitas bisnis secara elektronik yang efisien dan fleksibel. Pengertian *e-business* atau elektronik bisnis adalah kegiatan bisnis yang dilakukan secara otomatis serta semiotomatis yang dilakukan dengan menggunakan teknologi elektronik. *E-business* memungkinkan suatu perusahaan untuk dapat berhubungan dengan sistem pemrosesan data internal maupun eksternal secara lebih efisien serta fleksibel. Mekanismi *e-business* menjadikan *e-commerce* berada pada bagian dari kegiatan *e-business* yang memiliki mekanisme yang sama, namun *e-business* sendiri memiliki cakupan yang lebih luasa.

Munculnya *e-commerce* ini menjadikan kegiatan-kegiatan konvensional *commerce* cenderung ditinggalkan. Hal yang sangat membedakan antara *e-commerce* dan konvensional *commerce* terletak pada penerapan teknologi yang lebih dominan dalam melakukan dua hal utama:

1. Mengolah data mentah menjadi informasi yang dapat dimanfaatkan bersama oleh para pelaku bisnis dan konsumen; dan
2. Mendistribusikan data atau informasi tersebut secara cepat dan efisien ke seluruh komponen bisnis yang membutuhkan.

Dari beragam jenis aplikasi *e-commerce* yang ada, secara prinsip mekanisme kerjanya kurang lebih sama, seperti yang terlihat pada gambar 2.1.



**Gambar 2.1Sistem *E-Commerce***

**2.2 *E-Marketplace***

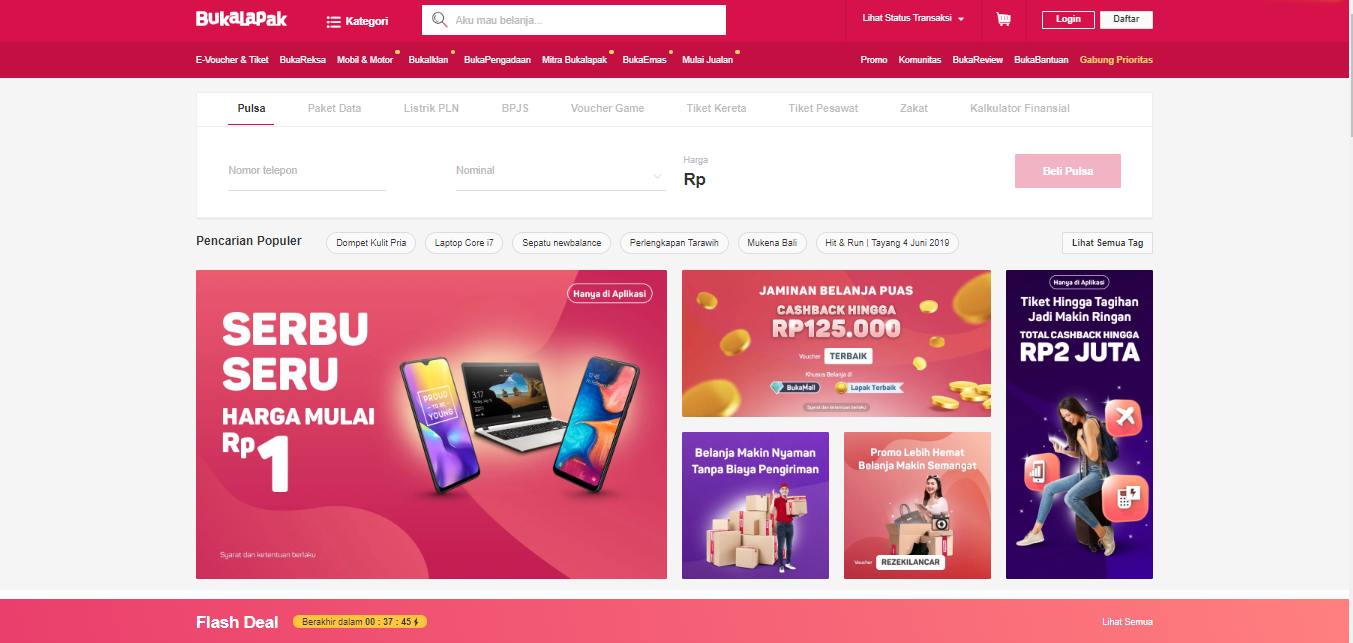
*E-marketplace* adalah sebuah pasar *virtual* dimana di dalamnya berkumpul para penjual dan pembeli yang menjalankan bisnisnya secara *online*. Layanan yang

ditawarkan oleh *e-marketplace* antara lain katalog elektronik yang menampilkan barang dan jasa, daftar direktori bisnis, dan lelang secara *online* (Soesanto, 2010).

Pada awalnya *e-marketplace* merupakan tempat bertemunya berbagai pihak atau entiti yang memiliki tujuan utama untuk berdagang (transaksi jual-beli). Produk atau jasa yang paling cocok untuk diperdagangkan dalam *e-marketplace* adalah yang bersifat komoditas. Sebab selain sesuai dengan karakteristik transaksi dagang yang cepat dan berjangka pendek, barang-barang komoditas sangat mudah untuk menentukan harga sehingga tidak sulit jika dipertukarkan secara global. Perbedaan yang sangat mendasar dari pasar konvensional terhadap *e-marketplace*  terletak pada konsep transparansi. Bagi *buyer* (pembeli) dapat melakukan perbandingan harga antara banyak *supliyer* barang yang sama, hal ini membuat pengusaha *e-marketplace* menjadikan harga barang sebagai penarik minat *buyer* (pembeli) Berikut adalah *e-marketplace* yang sering dijumpai di Indonesia:

1. Bukalapak

Bukalapak adalah *e-marketplace* yang menawarkan berbagai macam jenis produk, mulai dari elektronik, buku, mainan anak dan perlengkapan bayi, alat kesehatan dan produk kecasntikan, peralatan rumah tangga, serta perlengkapan traveling dan olahraga. Bukalapak didirikan di Singapura pada tahun 2010.

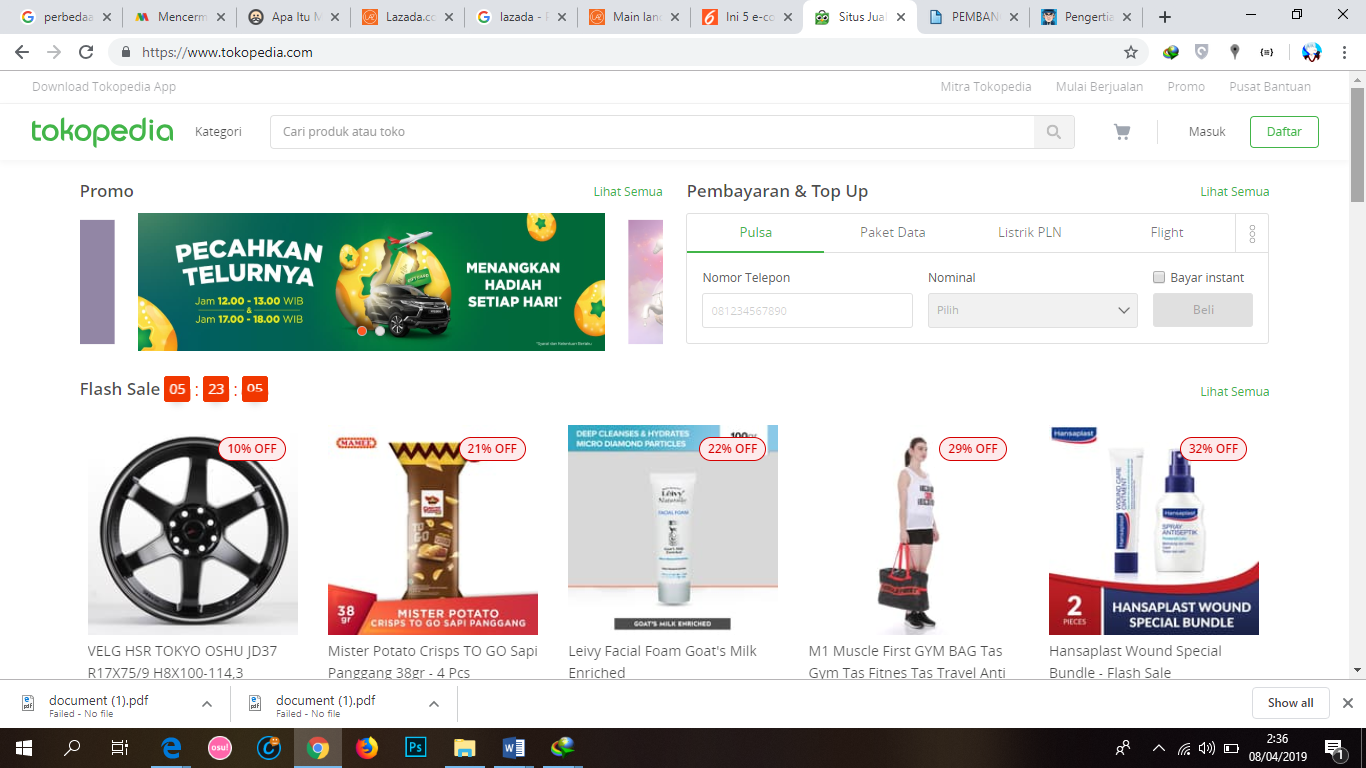


**Gambar 2.2 *E-marketplace* Bukalapak**

1. Tokopedia

Tokopedia merupakan perusahaan teknologi Indonesia dengan misi mencapai pemerataan ekonomi secara digital. Sejak didirikan pada tahun 2009, Tokopedia telah bertransformasi menjadi sebuah *unicorn* yang berpengaruh tidak hanya di Indonesia tetapi juga di Asia Tenggara.

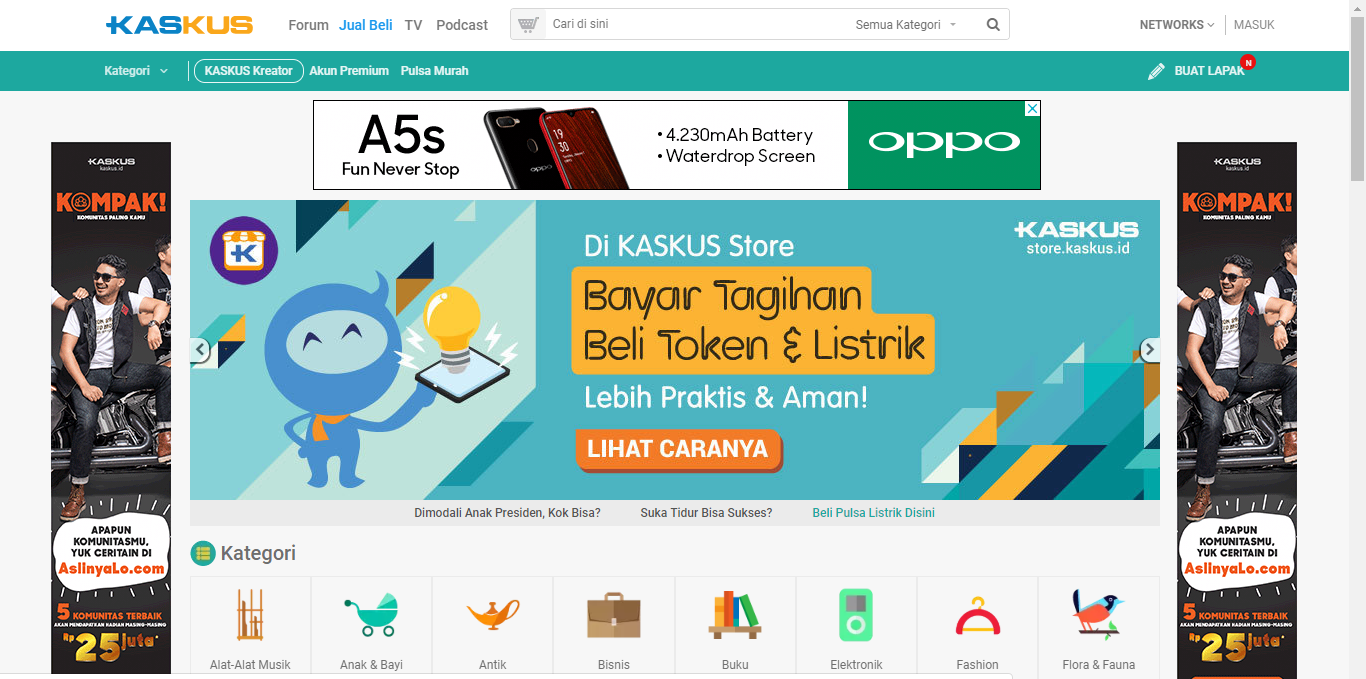
Tokopedia merupakan *e-marketplace* di Indonesia yang memungkinkan setiap individu, toko kecil, atau *brand* untuk membuka dan mengelola toko *online*. Hingga saat ini, Tokopedia menjadi *e-marketplace* yang paling banyak dikunjungi oleh masyarakat Indonesia. Sejak diluncurkan, layanan dasar Tokopedia dapat digunakan oleh semua orang secara gratis.



**Gambar 2.3 *E-marketplace* Tokopedia**

1. FJB Kaskus

FJB Kaskus merupakan forum jual beli dalam bisnis *online* yang cukup besar di Indonesia. Kaskus membuat sistem *escrow* yang diberi nama Rekening Bersaman atau Rekber. Rekber ini berfungsi sebagai pihak ketiga yang menyimpan uang saat melakukan transaksi.



**Gambar 2.4 *E-marketplace* FJB Kaskus**

**2.3 *Online* *Shop***

Online shop atau bisnis online saat ini bukan lagi menjadi sesuatu yang asing bagi masyarakat Indonesia, baik yang dalam kesehariannya menggunakan internet ataupun tidak. Adapun definisi online shop adalah suatu proses pembelian barang atau jasa melalui internet dimana antara penjual dan pembeli tidak pernah bertemu atau melakukan kontak secara fisik yang dimana barang yang ditawarkan melalui *display* dengan gambar yang ada di suatu website atau toko maya. Setelahnya pembeli dapat memilih barang yang diinginkan untuk kemudian melakukan pembayaran kepada penjual melalui rekening bank yang bersangkutan. Setelah proses pembayaran diterima, kewajiban penjual adalah mengirim barang pesanan pembeli ke alamat tujuan.

*Online shop* mengacu pada berbagai toko yang ada pada *platform online*. Pada dasarnya, online shop adalah tempat dimana terjadinya suatu transaksi penjualan barang atau jasa di internet. Online shop tidak harus ada pada website. Anda juga bisa membuka toko *online* di *social media* seperti Facebook dan Instagram.

Dalam memilih *platform* antara yang paling sesuai untuk seorang pengembang bisnis secara *online*, ada beberapa hal yang bisa menjadi pertimbangan, antara lain (Nathasya, 2018):

1. Faktor Teknis

Hal pertama yang menjadi pertimbangan adalah faktor teknis dan biaya. *E-commerce* tentunya akan membutuhkan lebih banyak biaya dan pengetahuan teknis dibanding *marketplace* atau *online shop* melalui *social media.* Kalau pebisnis membuat *website e-commerce*, pebisnis perlu membayar seseorang untuk membuat *website*-nya serta mengelolanya. Sedangkan *marketplace* dan *online shop,* pebisnis bisa meng-*upload* produk ke *platform* yang sudah ada. dan tidak perlu susah-susah memikirkan *design* dan biaya pembuatan *website*. Kebanyakan *marketplace* juga bersifat gratis sehingga pengguna hanya perlu daftar dan mengupload gambar Anda.

1. Efektifitas Biaya

Biaya untuk membuat *website* *e-commerce* sendiri akan memakan lebih banyak biaya jika dibanding dengan memanfaatkan *marketplace* dan *online shop* melalui *social media*. Tetapi ada faktor lain yang juga perlu dipertimbangkan. Jika menggunakan *marketplace* dan *social media*, maka persaingan akan lebih keras. Akan ada toko lain yang menawarkan barang yang sama dengan pengguna harus memasang harga yang bisa bersaing. Selain itu, lebih susah juga bagi pengguna untuk mendapat customer karena mereka memiliki banyak pilihan. Dengan *website e-commerce*, pebisnis bisa mendapat margin yang lebih tinggi. Kemungkinan besar, pengunjung website akan melakukan transaksi langsung di *website*.

1. Branding dan Reputasi

Tergantung pada apa yang dijual, menjual di *marketplace* dan *social media* bisa lebih bermanfaat. Misalnya, menjual baterai, maka tidak ada salahnya pebisnis menjual baterai di *marketplace* agar saat ada yang sedang belanja elektronik dan membutuhkan baterai, pengunjung bisa menemukan produk baterai yang dijual. Tetapi kalau pebisnis merasa bahwa produk tersebut unik dan merasa bahwa menjual barang itu di marketplace atau social media bisa memberi dampak kurang baik bagi penjualan dan reputasi, maka disarankan untuk membuat *website e-commerce*. Dengan *e-commerce*, pebisnis bisa meningkatkan brand *identity*. Pebisnis akan secara pribadi mengurus langsung pesanan pelanggan dan bertanggung jawab atas kepuasan mereka.

1. Marketing dan Traffic

Kalau pebisnis memasukkan produk ke *marketplace*, pebisnis tidak memiliki kontrol atas promosi yang mereka lakukan. Sebagai penyedia tempat, mereka ingin menjual sebanyak-banyaknya produk yang mereka bisa dan barang salah seorang pebisnis adalah salah satu dari ratusan barang yang ada di *website* *marketplace*. Penyedia akan melakukan berbagai strategi marketing dan promosi untuk mengundang orang-orang masuk ke *marketplace*. Dari segi *traffic*, barang pebisnis tentu akan dilihat banyak orang. Tetapi belum tentu mereka menjadi pelanggan, kalaupun mereka membeli barang dari toko seorang pebisnis di *marketplace*, pebisnis belum tentu bisa menjangkau pengunjung yang sama lagi.

**2.4 Algoritma Pencarian (*Search Algorithm*)**

Algoritma adalah [metode efektif](https://id.wikipedia.org/w/index.php?title=Metode_efektif&action=edit&redlink=1) diekspresikan sebagai rangkaian [terbatas](https://id.wikipedia.org/w/index.php?title=Terbatas&action=edit&redlink=1)dari instruksi-instruksi yang telah didefinisikan dengan baik untuk menghitung sebuah [fungsi](https://id.wikipedia.org/wiki/Fungsi_(matematika)). Dimulai dari sebuah kondisi awal dan input awal, instruksi-instruksi tersebut menjelaskan sebuah [komputasi](https://id.wikipedia.org/wiki/Komputasi) yang bila [dieksekusi](https://id.wikipedia.org/w/index.php?title=Eksekusi_(komputasi)&action=edit&redlink=1) akan diproses lewat sejumlah urutan kondisi terbatas yang terdefinisi dengan baik, yang pada akhirnya menghasilkan keluaran dan berhenti di kondisi akhir. Transisi dari satu kondisi ke kondisi selanjutnya tidak harus [deterministik](https://id.wikipedia.org/w/index.php?title=Deterministik&action=edit&redlink=1). Beberapa algoritma dikenal dengan [algoritma pengacakan](https://id.wikipedia.org/w/index.php?title=Algoritme_pengacakan&action=edit&redlink=1) menggunakan masukan acak.

Pencarian atau *searching* merupakan tindakan untuk mendapatkan suatu data dalam kumpulan data. Pencarian seringkali ditemukan di dalam kehidupan sehari-hari, misalnya untuk menemukan nomor telepon seseorang pada buku telepon atau mencari suatu istilah dalam kamus. Pada aplikasi komputer, pencarian biasanya dilakukan untuk mendapatkan data dari seorang mahasiswa, mendapatkan informasi suatu kata dalam kamus digital, mendapatkan nomor telepon berdasarkan suatu alamat atau nama perusahaan.

Algoritma pencarian atau *search algorithm* adalah algoritma yang menerima sebuah argumen kunci dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*successful*) atau tidak ditemukan (*unsuccessful*) (Sembiring, 2013).

Pencocokan *string* secara garis besar dibedakan menjadi dua yaitu *exact string matching* (pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan) dan *inexact string matching* (pencocokan *string* secara samar, yaitu pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan namun keduanya memiliki susunan karakter yang berbeda). Dua pendekatan inexact *string* matching meliputi *approximate* *string matching* yang mencocokkan *string* berdasarkan kemiripan penulisan dan *phonetic* *string matching* yang mencocokkan *string* berdasarkan kemiripan ucapan. *Approximate string matching* dapat digunakan untuk pencarian *string* berdasarkan *string* yang sama dan *string* yang memiliki kemiripan penulisan dengan *string* yang terdapat pada kamus.

**2.5 Algoritma *Levenshtein Distance***

*Levenshtein Distance* dibuat oleh Vladimir Levenshtein Distance pada tahun 1965. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*. *Levenshtein Distance* adalah sebuah matriks *string* yang digunakan untuk mengukur perbedaan atau jarak (*distance*) antara dua *string*. Nilai *distance* antara dua *string* ini ditentukan oleh jumlah minimum dari operasi-operasi perubahan yang diperlukan untuk melakukan

transformasi dari suatu *string* menjadi *string* lainnya. Operasi-operasi tersebut adalah penyisipan (*insertion*), penghapusan (*deletion*), atau penukaran (*subtition*). *Levenshtein Distance* merupakan salah satu algoritma yang dapat digunakan dalam mendeteksi kemiripan antara dua *string* yang berpotensi melakukan tindak plagiarisme (Purba & Situmorang, 2017).

Untuk menghitung *Levenshtein Distance* dari dua buah *string* dapat dilakukan secara rekursif. Formula dari algoritma rekursif untuk *string* a dan *string* b di berikan oleh 𝑙𝑒𝑣 𝑎, ( |𝑎| , |𝑏| ) dimana:

Misalkan i adalah indeks dari *string* a pada suatu tahap dan j adalah indeks *string* b. Formula yang diberikan di atas akan membandingkan *string* dari a dan *string* b dari indeks paling akhir. Untuk setiap tahap, akan diputuskan operasi mana yang lebih optimal. Untuk mengecek nilai solusi dari operasi penghapusan, maka status i akan dikurangi sejumlah satu. Untuk mengecek nilai solusi dari operasi penyisipan, maka status j akan dikurangi dengan sejumlah satu. Untuk operasi penggantian, maka status i dan status j masing-masing akan dikurangi dengan sejumlah satu. Bila karakter ke i dari *string* a dan karakter ke j dari *string* b sama, maka operasi penggantian tidak perlu diperhitungkan, tetapi status i dan status j dapat langsung dikurangi dengan jumlah satu.

Algoritma ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai cost. Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string.* Adapun contoh pencariannya sebagai berikut:

1. Melakukan *brute force* untuk setiap kata yang ada pada data.

2. Bila mendapatkan kalimat, maka kalimat akan dipisah pada setiap kalimatnya.

3. Mencocokkan kalimat yang ada dengan kata kunci yang telah dimasukkan oleh pengguna.

4. Melakukan perhitungan jarak kedua kata menggunakan algoritma *Levenshtein Distance*. Sebagai contoh perhitungan jarak antara kata “kamu” dan “kami”.

**Tabel 2.1 Inisialisasi Awal *Levenshtein Distance***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | - | K | A | M | I |
| - | 0 | 1 | 2 | 3 | 4 |
| K | 1 |  |  |  |  |
| A | 2 |  |  |  |  |
| M | 3 |  |  |  |  |
| U | 4 |  |  |  |  |

B­erdasarkan tabel 2.1, inisialisasi awal dilakukan dengan menyusun kata pertama secara vertikal dan kata kedua secara horizontal, lalu diberikan nilai secara berurutan pada setiap katanya.

**Tabel 2.2 Pencarian Baris Pertama *Levenshtein Distance***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | - | K | A | M | I |
| - | 0 | 1 | 2 | 3 | 4 |
| K | 1 | 0 | 1 | 2 | 3 |
| A | 2 |  |  |  |  |
| M | 3 |  |  |  |  |
| U | 4 |  |  |  |  |

Pada tabel 2.2 terdapat nilai “0” yang telah didapat dari pencarian nilai minimum dari nilai kotak diatasnya yang telah ditambahkan 1 dan disampingnya yang telah ditambahkan 1 beserta nilai kotak diatasnya yang ditambahkan 1 bila huruf yang dicocokkan bernilai sama.

**Tabel 2.3 Pencarian Baris Kedua *Levenshtein Distance***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | - | K | A | M | I |
| - | 0 | 1 | 2 | 3 | 4 |
| K | 1 | 0 | 1 | 2 | 3 |
| A | 2 | 1 | 0 | 1 | 2 |
| M | 3 |  |  |  |  |
| U | 4 |  |  |  |  |

Pada tabel 2.3 terbapat lanjutan baris yang dilakukan sama dengan cara sebelumnya. Dengan mencari nilai minimum dari beberapa nilai kotak.

**Tabel 2.4 Mendapatkan Nilai *Levenshtein Distance***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | - | K | A | M | I |
| - | 0 | 1 | 2 | 3 | 4 |
| K | 1 | 0 | 1 | 2 | 3 |
| A | 2 | 1 | 0 | 1 | 2 |
| M | 3 | 2 | 1 | 0 | 1 |
| U | 4 | 3 | 2 | 1 | 1 |

Pada tabel 2.4 terdapat hasil dari algoritma ini dengan mengisi semua kotak. Hasil dari algoritma ini dapat dilihat pada nilai akhir yang telah didapatkan yaitu “1”. Dengan ini menyatakan hasil dari algoritma *Levenshtein Distance* dari perbandingan kata “kamu” dan “kami” bernilai “1”.

Tahap selanjutnya memasukkan hasil *Levenshtein Distance* ke dalam similarity sebagai berikut:

**(2.2)**

Keterangan:

* Similarity = nilai kemiripan
* Dis = nilai *levenshtein distance*
* MaxLength = jumlah *string* terpanjang

Sim = 1 – () = 0.75

Dengan menggunakan rumus *similarity*, nilai kemiripan dari kata “KAMU” dan “KAMI” menghasilkan nilai 0.75.

**2.6 *Notasi Big-O***

Notasi menyatakan *running time* dari suatu algoritma untuk kemungkinan kasus terburuk. Notasi memiliki dari beberapa bentuk yaitu sebagai berikut (Azizah, 2013):

* Bentuk memiliki arti bahwa algoritma yang sedang dianalisis merupakan algoritma konstan. Hal ini mengindikasikan bahwa *running time* algoritma tersebut tetap, tidak bergantung pada .
* Bentuk berarti bahwa algoritma tersebut merupakan algoritma linier. Artinya, bila menjadi maka *running time* algoritma akan menjadi dua kali *running time* semula.
* Bentuk berarti bahwa algoritma tersebut merupakan algoritma kuadratik. Algoritma kuadratik biasanya hanya digunakan untuk kasus dengan yang berukuran kecil. Sebab, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi empat kali semula.
* Bentuk berarti bahwa algoritma tersebut merupakan algoritma kubik. pada algoritma kubik, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi delapan kali semula.
* Bentuk berarti bahwa algoritma tersebut merupakan algoritma eksponensial. Pada kasus ini, bila dinaikkan menjadi dua kali semula, maka *running time* algoritma akan menjadi kuadrat kali semula.
* Bentuk berarti algoritma tersebut merupakan algoritma logaritmik. Pada kasus ini, laju pertumbuhan waktu lebih lambat dari pada pertumbuhan *n*. Algoritma yang termasuk algoritma logaritmik adalah algoritma yang memecahkan persoalan besar dengan mentransformasikannya menjadi beberapa persoalan yang lebih kecil dengan ukuran sama. Basis algoritma tidak terlalu penting, sebab bila misalkan dinaikkan menjadi dua kali semula, meningkat sebesar jumlah tetapan.
* Bentuk , terdapat pada algoritma yang membagi persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan setiap persoalan secara independen, kemudian menggabungkan solusi masing-masing persoalan.
* Sedangkan bentuk berarti bahwa algoritma tersebut merupakan algoritma faktorial. Algoritma jenis ini akan memproses setiap masukan dan menghubungkannya dengan masukan lainnya. Bila menjadi dua kali semula, maka *running time* algoritma akan menjadi faktorial dari 2n.

**2.7 *Web Scraping***

*Web Scrapping* atau ekstraksi data *web* adalah suatu teknik untuk mengutip data atau informasi dari suatu *website* menggunakan *software* dengan program tertentu. Biasanya program dalam *software* tersebut mensimulasikan eksplorasi manusia terhadap suatu web dengan menggunakan *low-level* HTTP atau menggunakan *full-fledged web* tertentu seperti *Internet Explorer* atau *Mozilla*. *Web scrapping* berhubungan dengan pengindeksan *web* yang merupakan suatu teknik umum yang dipakai hampir semua *search engine*. Perbedaannya *web scraping* lebih berfokus pada transformasi dari suatu web yang tidak terstruktur, umumnya dalam format HTML menjadi suatu format data terstruktur yang dapat disimpan dan dianalisa pada *database* atau lembar kerja (Utomo, 2013).

*Web scrapping* juga terkait dengan otomasi web, yang mensimulasikan aktifitas web browsing dari manusia menggunakan perangkat lunak komputer. *Web scraping* menjadi semacam solusi praktis yang ada. Berikut ini beberapa level dari otomasi yang tersedia pada web scraping antara lain (Ryanda, 2015) :

- *Human copy-and-paste*: sering terjadi bahwa teknologi *web scraping* tidak bisa menggantikan manusia dari pemeriksaan manual dan mengcopy-paste, kadang-kadang hal ini dapat menjadi satu-satunya solusi yang ada ketika situs web secara eksplisit terdapat hambatan untuk mencegah mesin otomasi.

- *Text Grepping and Regular Expression Matchin*g: Sebuah pendekatan sederhana namun canggih untuk mengambil informasi dari halaman web berdasarkan unix grep perintah dan kalimat biasa cocok dengan menggunakan bahasa pemograman Perl atau Phyton.

- HTTP *programming*: statis dan dinamis halaman web dapat diambil dengan permintaan HTTP ke server *web* yang jauh (*remote*) menggunakan pemograman *socket*.

- DOM *parsing*: dengan menambahkan suatu *full-fledged web browser*, seperti *Internet Explorer* atau *Mozilla*, program dapat mengambil isi dinamis yang dihasilkan dari skrip pada sisi klien.

- HTML *parsers*: beberapa bahasa *query* data semi berstruktur, seperti *Hyper-Text Query Language* (HTQL) yang dapat digunakan untuk mem-*parsing* halaman HTML dan untuk mengambil konten dan mentransformasikan web

- *Web scraping software*: ada banyak perangkat lunak *web scraping software* yang dapat digunakan untuk solusi *web scraping*. Perangkat lunak tersebut mungkin menyediakan perekam *web* sehingga menghilangkan penulisan kode secara manual untuk *web scraping* yang dapat digunakan untuk mengekstrak isi web agar dapat menyimpan data ke *database* lokal.

Pada penelitian ini melakukan teknik DOM *parsing* dan HTML *parsers*, dimana data website akan diambil baik dari HTML maupun hasil javascript DOM yang telah berubah menjadi bentuk HTML. Pada HTML ini akan diambil data yang berhubungan dengan penelitian.

**2.8 Python**

Python adalah bahasa pemrograman *interpretatif* multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python merupakan salah satu bahasa pemrograman yang populer di dunia kerja Indonesia. Python juga merupakan bahasa pemrograman yang relatif mudah untuk pemula. Selain itu di akademik pun banyak akademisi yang menggunakan python untuk menyelesaikan penelitiannya di bidang robotika, data science, ekonomi, antariksa dan berbagai macam bidang lainnya. Python secara default telah terpasang di beberapa sistem operasi berbasis Linux seperti Ubuntu, Linux Mint, dan Fedora. Untuk sistem operasi lain, sudah tersedia installer yang disediakan untuk sistem operasi tersebut.

Python memiliki kelebihan tersendiri dibandingkan dengan bahasa pemrograman lainnya terutama dalam hal penanganan modul, ini yang membuat beberapa programmer menyukai Python. Selain itu Python merupakan salah satu produk yang bersifat *open* *source*, *free*, dan multiplatform. Beberapa *feature* yang dimiliki Python antara lain.

1. Memiliki *library* yang luas, dalam distribusi Python telah disediakan modul-modul siap pakai untuk berbagai keperluan
2. Memiliki tata bahasa yang jernih dan mudah dipelajari
3. Memiliki aturan layout *source code* yang memudahkan pengecekan, pembacaan kembali, dan penulisan ulang *source code* tersebut
4. Bersifat OOP
5. Memiliki sistem pengelolaan *memory* yang *auto*, *garbage* *collection*, seperti Java
6. Bersifat modular, mudah dikembangkan dengan menciptakan modul-modul baru, dimana modul-modul tersebut dapat dibangun dengan bahasa Python maupun C/C++

**2.9 Basis Data (*Database*)**

Basis data merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan sesuai struktur tertentu dan disimpan dengan baik.  Untuk mendapatkan informasi yang berguna dari kumpulan data, maka diperlukan suatu perangkat lunak (*software*) untuk memanipulasi data sehingga mendapatkan informasi yang berguna. *Database* terbentuk dari beberapa komponen, yaitu:

1. *Table*

*Table* atau Tabel adalah sekumpulan data dengan struktur yang sedemikian rupa, terbentuk dari *record* dan *field*.

1. *Record*

*Record* adalah sekumpulan *field* yang membentuk suatu objek tertentu

1. *Field*

*Field* adalah atribut dari objek yang memiliki tipe data tertentu.

Menurut jenisnya dapat digolongkan menjadi dua jenis basis data, antara lain:

1. Basis Data *Flat-File*

Basis data ini ideal untuk data berukuran kecil dan dapat dirubah dengan mudah sesuai dengan kebutuhan. Pada dasarnya, basis data *flat-file* tersusun dari sekumpulan *string* dalam satu atau lebih *file* yang dapat diurai untuk mendapatkan informasi yang disimpan. Basis data *flat-file* cocok untuk menyimpan daftar atau data yang sederhana dan dalam jumlah kecil. Basis data *flat-file* akan menjadi sangat rumit apabila digunakan untuk menyimpan data dengan struktur kompleks walaupun dimungkinkan pula untuk itu.

Beberapa kendala dalam menggunakan basis data jenis ini adalah rentan pada korupsi data karena tidak adanya penguncian yang melekat ketika data digunakan atau dimodifikasi dan juga adanya duplikasi data yang mungkin sulit dihindari. Salah satu tipe basis data flat-file adalah file CSV yang menggunakan pemisah koma untuk setiap nilainya.

1. Basis Data *Relational*

Basis data ini mempunyai struktur yang lebih logis terkait cara penyimpanan. Kata "relasional" berasal dari kenyataan bahwa tabel-tabel yang ada di basis data relasional dihubungkan satu dengan lainnya. Basis data relasional menggunakan sekumpulan tabel dua dimensi yang masing-masing tabel tersusun atas baris (tupel) dan kolom (atribut).

Untuk membuat hubungan antara dua atau lebih tabel, digunakan key (atribut kunci) yaitu primary key di salah satu tabel dan foreign key di tabel yang lain. Saat ini, basis data relasional menjadi pilihan utama karena keunggulannya. Program aplikasi untuk mengakses basis data relasional menjadi lebih mudah dibuat dan dikembangkan dibandingkan dengan penggunaan basis data *flat-file*.

**2.10 *Flowchart***

*Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program (Pahlevy. 2010). Flowchart menolong analis dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

*Flowchart* merupakan simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung sehingga setiap simbol *flowchart* melambangkan pekerjaan dan instruksinya. Simbol-simbol yang digunakan dalam *flowchart* adalah sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabel 2.5 Simbol-simbol *flowchart* (Ewolf, 2011)** | | | | |
| **NO.** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1. |  | Proses | Mempresentasikan operasi. |
| 2. |  | *Input / Output* | Mempresentasikan *Input* atau *Output* data yang diproses atau informasi. |
| 3. |  | Keputusan | Keputusan dalam program. |
| 4. |  | Dokumen | Dokument I / O dalam format cetak. |
| 5. |  | *Terminal points* | Awal / akhir *flowchart*. |
| 6. |  | *Preparation* | Pemberian harga awal. |
| 7. |  | Manual *input* | *Input* yang dimasukkan secara manual dari keyboard. |
| 8. |  | Penghubung | Keluar atau masuk dari bagian lain *flowchart* khususnya |
| 9. |  | Penghubung | Keluar atau masuknya dari bagian lain *flowchart*  khususnya halaman lain. |
| 10. |  | *Display* | *Output* yang ditampilkan pada terminal |
| 11. |  | Anak panah | Mempresentasikan alur kerja. |

**2.11 *Unified Modeling Language* (UML)**

*Unified Modeling Language* (UML) merupakan bahasa dalam mendesain perangkat lunak secara visual. Dengan UML, desainer dapat melihat konsep global suatu desain. Desain kemudian dapat dijadikan panduan dalam proses pengembangan dan rekayasa perangkat lunak. Selain itu, UML dapat menjadi media komunikasi gagasan antara pengembang perangkat lunak dengan pengguna (Shalahudin, 2011). Ada beberapa jenis diagram dalam UML yaitu:

1. ***Use Case Diagram***

*Use case diagram* menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam sistem (Shalahudin, 2011).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.6 Simbol *Use Case Diagram*** | | | |
| **NO** | | **GAMBAR** | **NAMA** | **KETERANGAN** | |
| 1. | |  | *Actor* | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case.* | |
| 2. | |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*dependent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). | |
| 3. | |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*). | |

**Tabel 2.6 Lanjutan**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 4. |  | *Include* | Menspesifikasikan bahwa *use case* sumber secara eksplisit. |
| 5. |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7. |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8. |  | *Use case* | Deskripsi dari uraian aksi-aksi yang ditampilkan system yang menghasilkan suatu hasil yang terukur bagi suatu *actor.* |
| 9. |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |

1. ***Activity Diagram***

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabel 2.7 Simbol *Activity Diagram*** | | | | | | |
| **NO** | | **GAMBAR** | | **NAMA** | | **KETERANGAN** | |
| 1. | |  | | *Activity* | | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. | |
| 2. |  | | *Action* | | State dari sistem yang mencerminkan eksekusi dari suatu aksi. | | |
| 3. |  | | *Initial Node* | | Bagaimana objek dibentuk atau diawali. | | |
| 4. |  | | *Activity Final Node* | | Bagaimana objek dibentuk dan diakhiri. | | |

1. ***Sequence Diagram***

*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu objek dan pesan yang dikirimkan dan diterima antar objek. Jumlah *Sequence diagram* yang akan digambar adalah minimal sama dengan jumlah *use case* yang didefinisikan (Shalahudin, 2011).

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.8 Simbol S*equence Diagram*** | | | |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** | |
| 1 |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. | |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. | |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. | |

**Tabel 2.8 Lanjutan**

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 4 |  | *Message* | Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy.* |

1. ***Class Diagram***

*Class diagram* adalah kumpulan objek-objek yang mempunyai struktur umum, *behavior* umum, relasi umum, dan *semantic* atau kata yang umum. *Class-class* ditentukan atau ditemukan dengan cara memeriksa objek-objek dalam *sequence diagram* dan *collaboration diagram*. Sebuah *class* digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabel 2.9 Simbol *Class Diagram*** | | | |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** | |
| 1. |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya | |
| 2. |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. | |
| 3. |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. | |
| 4. |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. | |
| 5. |  | *Dependency* | Perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. | |

**2.12 Metode Pengembangan Sistem**

Metode Pengembangan sistem yang digunakan pada tugas ini adalah metode *waterfall*. *Waterfall* sering juga disebut *sekuensial linier* atau alur hidup klasik. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut mulai dari analisis kebutuhan, desain sistem, pengodean, pengujian, dan tahapan pendukung (support). Berikut adalah model gambar metode *waterfall*.



**Gambar 2.5 Tahap-Tahap Metode *Waterfall***

1. Analisa Kebutuhan

Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau studi literatur. Sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menerjemahkan ke dalam bahasa pemrograman.

1. Desain Sistem

Tahapan dimana dilakukan penuangan pikiran dan perancangan sistem terhadap solusi dari permasalahan yang ada dengan menggunakan perangkat pemodelan sistem seperti diagram alur data (*data flow diagram*), diagram hubungan entitas (*entity relationship diagram*) serta struktur dan bahasan data.

1. Penulisan Kode Program

Penulisan kode program atau coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh programmer yang akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan ini lah yang merupakan

tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka

akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah

menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bias diperbaiki.

1. Pengujian Program

Tahapan akhir dimana sistem yang baru diuji kemampuan dan keefektifannya sehingga didapatkan kekurangan dan kelemahan sistem yang kemudian dilakukan pengkajian ulang dan perbaikan terhadap aplikasi menjadi lebih baik dan sempurna.

5. Penerapan Program dan Pemeliharaan

Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (*periperal* atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.