**PROPOSAL**

**IMPLEMENTASI *LEVENSHTEIN DISTANCE* PADA APLIKASI PENCARIAN BARANG DI BERBAGAI *E-MARKETPLACE* MENGGUNAKAN TEKNIK *WEB SCRAPING***

Diajukan Untuk Memenuhi

Salah Satu Syarat Memperoleh Gelar Sarjana Teknik



**AHMAD KHAIRUN ARSYAD**

**E1E1 15 054**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS HALU OLEO**

**KENDARI**

**2019**

**DAFTAR ISI**

**HALAMAN JUDUL i**

**DAFTAR ISI xii**

**DAFTAR GAMBAR . xv**

**DAFTAR TABEL xvii**

**BAB I PENDAHULUAN**  1

1.1 Latar Belakang 3

1.2 Rumusan Masalah 3

1.3 Batasan Masalah 3

1.4 Tujuan Penelitian 3

1.5 Manfaat Penelitian 3

1.6 Sistematika Penulisan 4

1.7 Tinjauan Pustaka 5

**BAB II LANDASAN TEORI**  7

2.1 *E-Commerce* 7

2.2 *E-Marketplace* 8

2.3 *Online Shop* 10

2.4 Algoritma Pencarian (*Search Algorithm*) 12

2.5 Metode *Levenshtein Distance* 13

2.6 *Notasi Big-O* 16

2.7 *Web Scraping* 17

2.8 Python 19

2.9 Basis Data *(Database)* 20

2.10 *Flowchart* 21

2.11 *Unified Modeling Language* (UML) 23

2.12 Metode Pengembangan Sistem 27

**BAB III METODOLOGI PENELITIAN**  29

3.1 Waktu dan Tempat Penelitian 29

3.3.1 Waktu Penelitian 29

3.3.2 Tempat Penelitian 30

3.2 Metode Pengumpulan Data 30

3.3 Metode Pengembangan Sistem 30

3.4 Analisis Sistem 33

3.4.1 Analisis Kebutuhan Fungsional 33

3.4.2 Analisis Kebutuhan Non-Fungsional 33

3.4.2.1 Kebutuhan Perangkat Keras 34

3.4.2.2 Kebutuhan Perangkat Lunak 34

3.5 Analisis Perancangan Sistem 35

3.5.1 Perancangan *Flowchar*t 35

3.5.1.1 *Flowchart* Sistem 35

3.5.1.2 *Flowchart* metode *Levenshtein Distance* 36

3.5.2 *Unified Modeling Language* (UML) 39

3.5.2.1 *Use Case Diagram* 39

3.5.2.2 *Activity Diagram* 41

3.5.2.3 *Class Diagram* 45

3.5.2.3 *Sequence Diagram* 46

3.6 Perancangan *User Interface* 47

3.6.1 Rancangan Beranda 47

3.6.2 Rancangan *Form* Pencarian 48

3.6.3 Rancangan Menu Petunjuk 48

3.6.4 Rancangan Menu Tentang 49

3.6.5 Rancangan *Form* Login 49 3.6.6 Rancangan Menu Halaman Admin 50

3.6.7 Rancangan Menu Cari Data 50

3.6.8 Rancangan Menu Lihat Data 51

3.6.9 Rancangan Menu Lihat Pengguna 51

3.7 Metode Pengujian 52

3.8 Rencana Pengujian 52

**DAFTAR PUSTAKA**

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Perkembangan teknologi informasi yang sangat pesat saat ini, mengakibatkan perkembangan pengguna internet di Indonesia tumbuh semakin meningkat. Pada tahun 2017 pengguna internet di Indonesia meningkat 8% menjadi 143, 26 juta pengguna. Jika dibandingkan dengan data populasi penduduk Indonesia yang sebanyak 262 juta jiwa, maka pengguna internet di Indonesia sudah mencapai 54,68% dari total populasi di Indonesia. Perkembangan ini membawa pengaruh terhadap berbagai aspek peradaban, mulai dari sosial, pendidikan, kesehatan, administrasi, dan masih banyak lagi (Tim APJII, 2018).

Dari sekian banyak aspek, ekonomi merupakan aspek yang sangat dipengaruhi oleh peningkatan penggunaan internet. Dengan banyaknya penduduk Indonesia yang menggunakan internet, membuat model pertumbuhan ekonomi yang semula masih konvensional kini semakin mengarah kepada perekonomian yang berbasis internet. Salah satu bentuk pemanfaatan internet terhadap aktifitas perekonomian adalah toko *online (online shop).* *Online shop* atau yang familiar disebut *olshop* hadir ditengah masyarakat didasarkan dari kecendrungan masyarakat dalam pemanfaatan internet pada aspek kehidupan. Bahkan saat ini hampir semua bentuk kegiatan jual beli dapat dilakukan hanya dengan menggunakan perangkat elektronik dan internet. Sehingga saat ini transaksi jual beli konvensional mulai tergeser dengan transaksi jual beli modern, maka muncullah istilah perdagangan elektronik yang biasa disebut *electronic commerce (e-commerce).*

*E-commerce* sangat menguntungkan bagi produsen dan juga sangat memudahkan bagi konsumen, karena sistem pelayanan melalui internet sangat efisien, praktis dan dapat menghemat waktu untuk memasarkan produk atau jasa. Kemudahan ini menyebabkan hadirnya berbagai macam toko yang tidak memiliki bentuk fisik yaitu toko-toko maya di dunia internet atau yang dewasa ini disebut *electronic marketplace (e-marketplace)*.

*E-marketplace* yang kian merebak di dunia maya membuat persaingan untuk mendapatkan pengunjung menjadi meningkat. Beberapa *e-marketplace* mengandalkan potongan harga atau menawarkan barang dagangan dengan harga yang murah untuk menarik minat pengunjung. Namun dengan kian banyaknya *e-marketplace* yang ada saat ini membuat pengunjung kesulitan dalam menentukan atau membandingkan suatu barang yang sama pada *e-marketplace* yang berbeda.

Agar memudahkan pencarian barang pada sebuah *e-marketplace*, biasanya terdapat sistem pencarian atau mesin pencarian pada laman *website* *e-marketplace* yang dapat memfilter data sesuai dengan kata kunci yang dimasukkan oleh pengunjung. Penggunaan mesin pencarian dianggap lebih efisien dan cepat dibandingkan dengan memberikan menu *list* kepada pengunjung untuk menentukan barang yang dituju. Namun untuk memadukan data dari berbagai *website* *e-marketplace* yang ada, dapat dilakukan dengan menggunakan teknik *Web Scraping*. *Web Scraping* sendiri adalah pengambilan sebuah dokumen semi terstruktur dari berbagai sumber, yang umumnya berupa halaman-halaman web dalam bahasa markup, yang kemudian dianalisis untuk mengambil data tertentu. Dengan adanya sistem pencarian menggunakan teknik *web scraping*, *search engine* dapat melakukan pencarian *string* *input*-an terhadap beberapa *website* sekaligus.

Dalam menyempurnakan sebuah sistem pencari yang dapat memuat data dari berbagai sumber dibutuhkan sebuah algoritma yang dapat menganalisis *string*. *Levenshtein Distance* merupakan sebuah algoritma yang dapat mengukur kesamaan antara *string*. Penggabungan antara teknik *Scraping* dan algoritma *Levenshtein Distance* ini yang nantinya akan membantu pengunjung dalam menemukan dan membandingkan barang yang diinginkan.

Melihat permasalahan diatas, maka peneliti mengembangkan sebuah sistem yang mampu melakukan pencarian barang terhadap beberapa *e-marketplace* yang ada, agar dapat melakukan perbandingan atas barang yang sama terhadap *e-marketplace* yang berbeda. Atas dasar inilah peneliti mengangkat judul ***“*Implementasi *Levenshtein Distance* pada Aplikasi Pencarian Barang di Berbagai *E-Marketplace* Menggunakan Teknik *Web Scraping”*.**

* 1. **Rumusan Masalah**

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menggunakan teknik *web scraping* agar mendapatkan data barang pada Bukalapak, Tokopedia dan FJB Kaskus?
2. Bagaimana mengimplementasikan algoritma *Levenshtein Distance* beserta kompleksitasnya?
3. Bagaimana cara mengujian pada sistem pencarian barang?
   1. **Batasan Masalah**

Adapun hal-hal yang menjadi batasan masalah pada penelitian ini adalah sebagai berikut :

1. Menggunakan Algoritma *Levenshtein Distance* sebagai algoritma pencarian barang pada aplikasi pencarian di berbagai *e-marketplace*.
2. Fungsi pencarian barang tidak dari semua *e-marketplace* tetapi hanya mengumpulkan dari Bukalapak, Tokopedia dan FJB Kaskus.
3. Data dari *e-marketplace* akan disimpan kedalam *database*.
4. Dibangun berbasis *website*.
   1. **Tujuan Penelitian**

Berdasarkan rumusan masalah yang sudah disebutkan sebelumnya, maka tujuan dari penelitian ini adalah untuk mengimplementasikan *Levenshtein Distance* pada aplikasi pencarian barang di berbagai *e-marketplace* menggunakan teknik *web scraping*.

* 1. **Manfaat Penelitian**

Manfaat yang diharapkan dari penelitian ini dapat memberi keuntungan bagi pihak pembeli dan penjual. Dengan pencarian ini, pembeli dapat dengan mudah melakukan pencarian barang di berbagai *e-marketplace* tanpa harus membuka banyak *e-marketplace* secara langsung. Untuk keuntungan penjual, tidak perlu lagi memasang produk yang sama di beberapa *e-marketplace*.

* 1. **Sistematika Penulisan**

Sistematika penulisan tugas akhir digunakan untuk menjelaskan penulisan per bab sebagai berikut:

**BAB I PENDAHULUAN**

Merupakan bab pendahuluan yang menguraikan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

**BAB II LANDASAN TEORI**

Membahas mengenai dasar-dasar teori pendukung yang berhubungan dengan masalah yang diambil dan program aplikasi yang akan digunakan dalam pembangunan sistem.

**BAB III METODE PENELITIAN**

Membahas mengenai waktu dan tempat penelitian, metode pengumpulan data, metode pembangunan sistem yang digunakan dalam penelitian tugas akhir ini.

**BAB IV ANALISIS DAN PERANCANGAN**

Bab ini akan membahas tentang analisis dan perancangan dari aplikasi yang akan dibuat, dengan menggunakan desain UML (*Unified Modelling Language*) serta desain *user interface*.

**BAB V IMPLEMENTASI DAN PEMBAHASAN**

Dalam bab ini akan dikaji mengenai implementasi hasil perancangan aplikasi yang dibuat serta melakukan pengujian terhadap sistem.

**BAB VI PENUTUP**

Bab ini berisi kesimpulan dari program yang telah dibuat serta saran yang diperlukan untuk pengembangan program berikutnya.

* 1. **Tinjauan Pustaka**

Penelitian yang dilakukan oleh Ariyani, dkk (2016) menyimpulkan bahwa pengujian menggunakan data *real* yaitu data dokumen berplagiat yang diambil dari artikel/berita lewat internet, Algoritma *Levenshtein Distance* menghasilkan nilai *similarity* yang tinggi yaitu diatas 85 % sampai 100 % untuk dokumen yang tingkat kemiripannya tinggi. Sedangkan untuk dokumen dengan tingkat kemiripan yang rendah atau tidak berplagiat maka menghasilkan nilai *similarity* dibawah 40%.

Pratama dan Pamungkas (2016) menyimpulkan bahwa Algoritma *Levenshtein Distance* untuk menghitung jumlah perbedaan antara dua *string*. Perhitungan jarak antara dua *string* ini ditentukan dari jumlah minimum operasi perubahan untuk mendeteksi kemiripan suatu dokumen teks dengan dokumen teks lainnya dengan cara pembentukan suatu matriks *string* untuk mendapatkan nilai *distance*, kemudian melakukan perhitungan bobot *similarity* antar dokumen teks berdasarkan nilai *distance* tersebut. Pada proses *sorting* juga menjadi sangat berpengaruh bagi algoritma *Levenshtein distance* dalam mendeteksi kemiripan dokumen teks. Dengan kata lain, Algoritma *Levenshtein distance* akan bekerja lebih baik jika kedua dokumen yang dibandingkan mempunyai urutan kata yang sama. Selain itu penggunaan proses *stopword*, *stemming*, dan *sorting* sekaligus dapat memberikan hasil yang lebih spesifik. Artinya, dalam mendeteksi kemiripan dokumen teks, akan lebih baik jika dilakukan pengeliminasian kata-kata yang umum, dan megubah kata berimbuhan menjadi kata dasar.

Panjaitan (2018) menyimpulkan bahwa dengan melakukan implementasi *Web Scraping* dapat mengumpulkan data melalui *website* dimana metode ini melakukan ekstraksi elemen *tag website* menjadi data yang dapat disimpan ke dalam database, sehingga dapat membentuk sebuah model pengumpulan data cuaca yang *termonitoring* dengan baik dan efisien. Identifikasi keadaan cuaca menjadi lebih akurat dengan memanfaatkan data yang besar dari hasil *scraping* dengan dikombinasikan dengan *K-Nearest Neighbor*. *K-Nearest Neighbor* dapat melakuan *resampling* dari data *historis* yang diaplikasikan untuk menentukan kondisi cuaca berdasarkan skenario cuaca yang memungkinkan. Sehingga menghasilkan akurasi identifikasi sebesar 98%.

Berdasarkan penelitian-penelitian yang telah dipaparkan di atas, maka akan dilakukan penelitian lebih lanjut dengan judul **“Implementasi *Levenshtein Distance* pada Aplikasi Pencarian Barang di Berbagai *E-Marketplace* Menggunakan Teknik *Web Scraping”.***

**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***

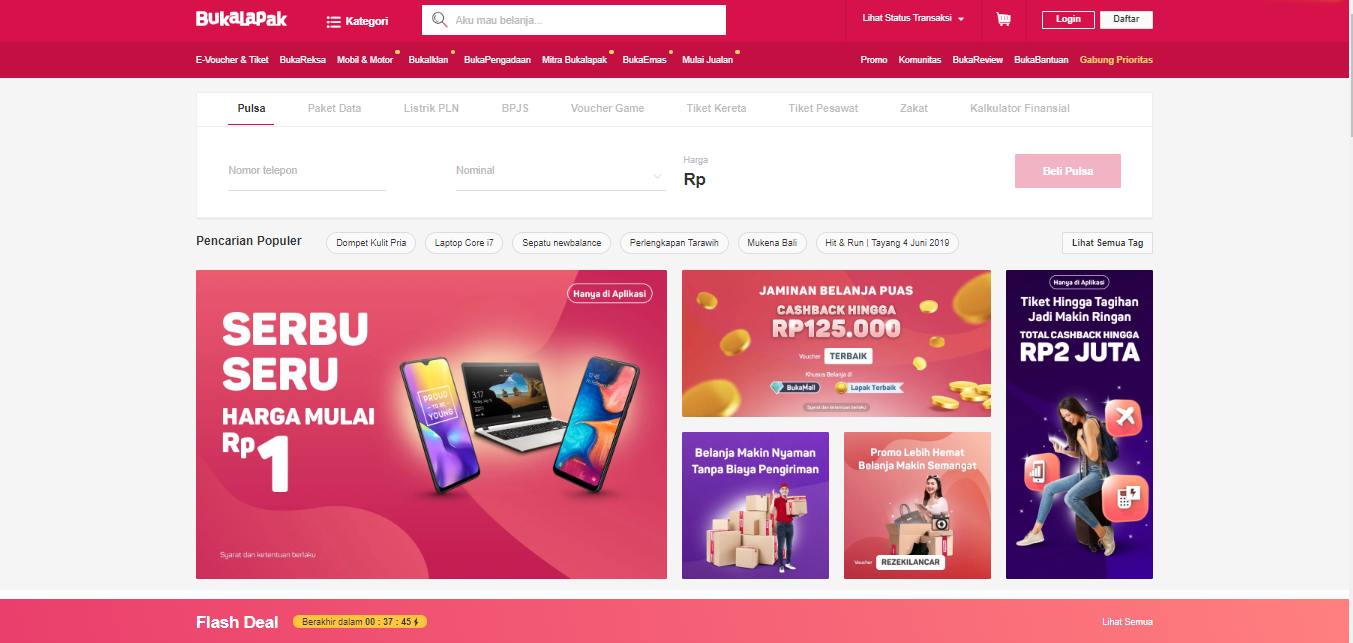
*E-marketplace* adalah sebuah pasar *virtual* dimana di dalamnya berkumpul para penjual dan pembeli yang menjalankan bisnisnya secara *online*. Layanan yang

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.

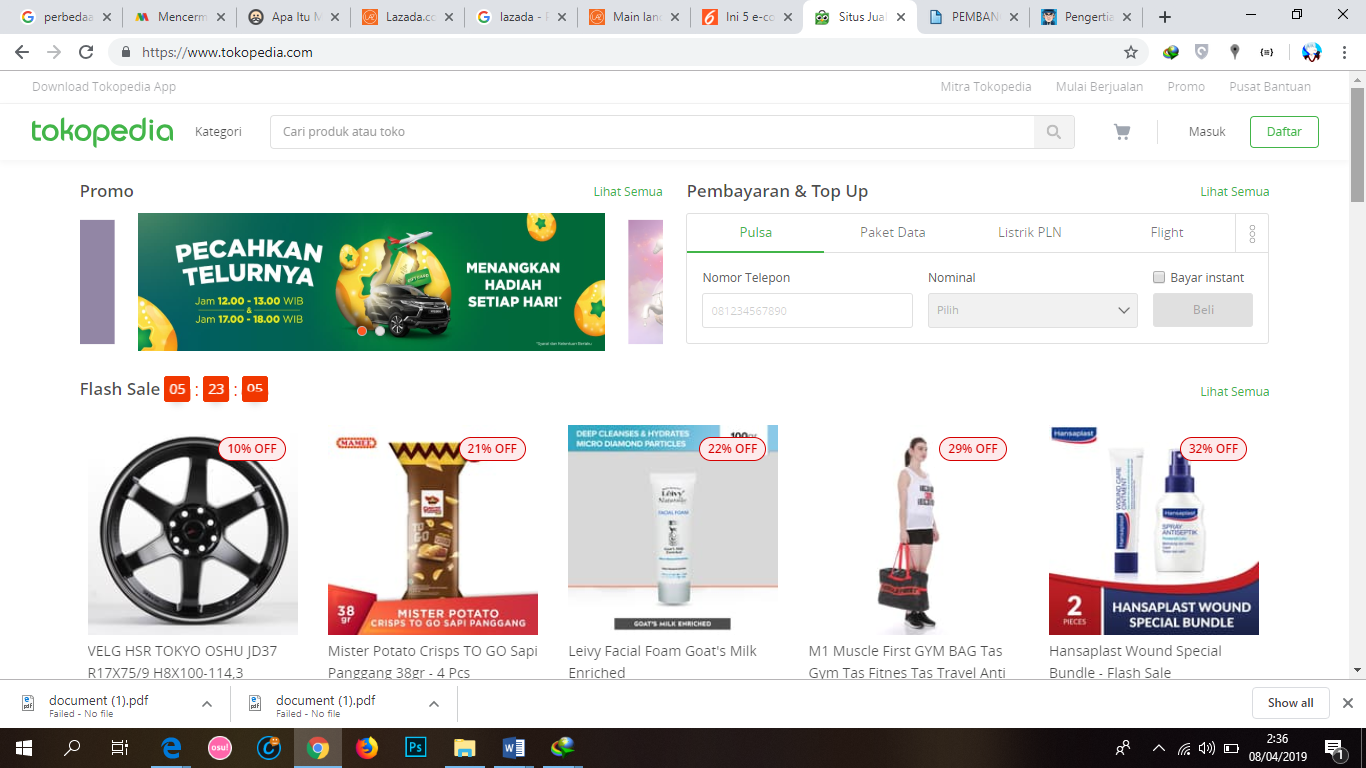


**Gambar 2.2 *E-marketplace* Bukalapak**

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.

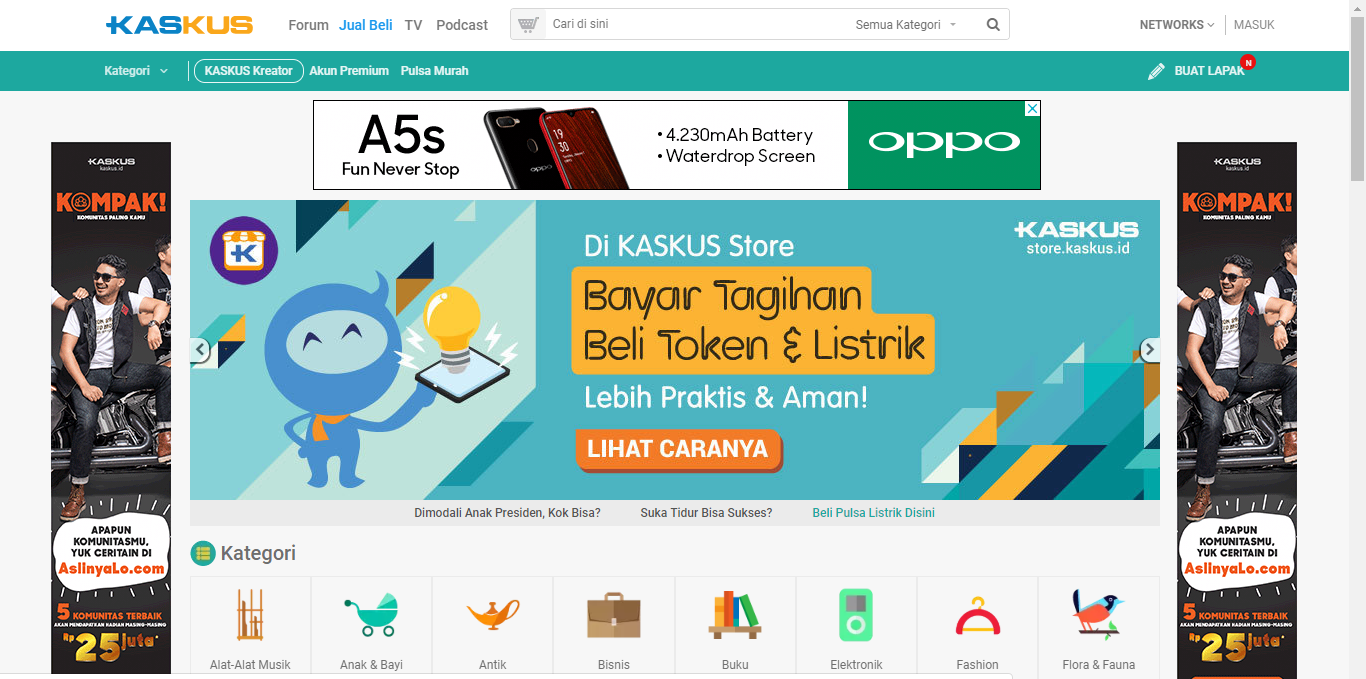
Tokopedia merupakan *e-marketplace* di Indonesia yang memungkinkan setiap individu, toko kecil, atau *brand* untuk membuka dan mengelola toko *online*. Hingga saat ini, Tokopedia menjadi *e-marketplace* yang paling banyak dikunjungi oleh masyarakat Indonesia. Sejak diluncurkan, layanan dasar Tokopedia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

**BAB III**

**METODOLOGI PENELITIAN**

**3.1 Waktu dan Tempat Penelitian**

**3.1.1 Waktu Penelitian**

Waktu pelaksanaan penelitian tugas akhir dilaksanakan mulai dari bulan Februari 2019 sampai dengan Juli 2019. Rincian kegiatan dapat dilihat pada Tabel 3.1 berikut:

**Tabel 3.1** ***Gantt Chart* Waktu Penelitian**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Uraian | Waktu (2019) Bulan Ke | | | | | | | | | | | | | | | | | | | | |
| II | | | III | | | | IV | | | | V | | | | VI | | | | VII | |
| 3 | 4 | 1 | | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 |
| 1 | Studi Literatur |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Analisis |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Desain |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Pembuatan Program |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Pengujian |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Pemeliharaan |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Dari *gantt chart* diatas diketahui lama pengerjaan selama 20 minggu terhitung sejak minggu ke-3 bulan Februari sampai dengan minggu ke-2 bulan Juli dengan penjabaran sebagai berikut:

1. Studi literatur dilakukan selama 5 minggu yaitu pada minggu ke-3 dan ke-4 bulan Februari dan dilanjutkan pada bulan Maret sampai minggu ke-3. Studi literatur dilakukan dengan cara membaca artikel, buku, situs dan referensi lainya yang berkaitan dengan pembuatan sistem.
2. Analisis dilakukan selama 3 minggu yaitu pada minggu ke-4 bulan Maret dan dilanjutkan sampai minggu ke-2 bulan April. Analisis dilakukan dengan menganalisis aplikasi yang akan dibuat dengan melihat permasalahan yang terjadi.
3. Desain dilakukan selama 4 minggu yaitu pada minggu 3 bulan April sampai dengan minggu ke-2 bulan Mei. Desain dilakukan dengan memperhatikan hasil perancangan sebelumnya.
4. Pembuatan program dilakukan pada minggu selanjutnya setelah desain dilakukan yaitu selama 4 minggu, dimulai dari minggu ke-3 bulan Mei sampai dengan minggu ke-2 bulan Juni.
5. Pengujian dimulai pada minggu terakhir pembuatan program selama 3 minggu sampai dengan minggu terakhir bulan juni. Tahap ini dilakukan untuk memastikan kesesuaian sistem yang telah dibangun terhadap rancangan awal
6. Pemeliharaan dilakukan dalam 2 minggu terakhir dimulai dari minggu pertama bulan Juli sampai dengan minggu ke-2 bulan Juli. Pemeliharaan dilakukan untuk memastikan sistem tidak menemui masalah saat digunakan.

**3.1.2 Tempat Penelitian**

Adapun tempat penelitian tugas akhir ini adalah laboratorium *Computer System And Networking* pada Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo.

**3.2 Metode Pengumpulan Data**

Dalam penelitian ini metode pengumpulan data yang digunakan pada perancangan aplikasi tersebut adalah studi literaturdan pengambilan data barang dari *e-marketplace* Tokopedia, Bukalapak dan FJB Kaskus menggunakan teknik *web scraping*. Metode studiliteratur ini dilaksanakan dengan melakukan studi kepustakaan yang relevan. Metode ini dilakukan untuk mencari penelitian yang sejenis dengan penelitian ini.

**3.3 Metode Pengembangan Sistem**

Metode pengembangan sistem yang digunakan dalam sistem ini adalah metode *Waterfall*. Dalam metode ini, terdapat lima tahap pengembangan perangkat lunak, yaitu:

1. Analisis

Pada tahap analisis adalah tahap dimana peneliti menganalisis aplikasi yang akan dibuat dengan melihat permasalahan yang terjadi. Permasalahan pengaruh internet dalam perekonomian di Indonesia. Pada tahun 2017 pengguna internet di Indonesia meningkat 8% menjadi 143, 26 juta pengguna. Jika dibandingkan dengan data populasi penduduk Indonesia yang sebanyak 262 juta jiwa (Tim APJII, 2018), dengan banyaknya pengguna internet di Indonesia, membuat model pertumbuhan ekonomi yang semula masih konvensional kini semakin mengarah kepada perekonomian yang berbasis internet. Sehingga saat ini transaksi jual beli konvensional mulai tergeser dengan transaksi jual beli modern yaitu *e-marketplace.* Dengan perkembangan *e-marketplace*, peneliti mengembangkan sistem pencarian barang dalam beberapa *e-marketplace.* Adapun *e-marketplace* yang dimaksud adalah Tokopedia, FJB Kaskus dan Bukalapak.

1. Desain

Pada tahap desain merupakan tahap dimana peneliti merancang pembuatan tampilan awal atau *interface* dari aplikasi yang akan dibangun. Tampilan *interface* akan dirancang melalui data-data yang telah diperoleh pada saat tahap analisis. Tampilan yang akan dibuat adalah beranda yang berperan sebagai *interface*/laman awal saat membuka sistem search market, *form* pencarian sebagai mesin pencarian bagi pengguna untuk menemukan barang yang di inginkan, menu petunjuk yang berfungsi sebagai instruksi cara penggunaan sistem search market, menu tentang sebagai informasi tambahan tentang pembuat sistem search market, *form login* sebagai laman untuk admin masuk ke halaman admin search market, halaman *dashboard* admin sebagai *interface* awal admin saat berhasil *login*, dan yang terakhir *form* pencarian data dan lihat semua data.

1. Pembuatan Program

Pada tahap pembuatan program yaitu penerjemahan design dalam bahasa yang bisa dikenali oleh komputer. Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam sistem search market ini menggunakan Bahasa pemrograman python. Penggunaan bahasa pemrograman python dikarenakan dalam *permormance load browser* dan *web server* lebih cepat dalam merespon permintaan *client* dibandingkan dengan menggunakan Bahasa pemrograman php (Winardi, Abdillah, Hermanto, & Widiyanto, 2018).

1. Pengujian

Pengujian merupakan tahap pengujian terhadap sistem yang telah dibuat menjadi suatu perangkat lunak yang siap pakai dengan tujuan untuk mengetahui apakah sistem yang dibuat sudah sesuai dengan yang diharapkan. Pengujian dilakukan dengan cara mencoba semua fungsi yang telah dibuat. Jika fungsi tidak berjalan dengan baik, maka proses pembuatan fungsi tersebut akan diulang dari tahap pembuatan program. Pengujian juga dilakukan terhadap *user* (pengguna sistem) dengan menggunakan kuesioner*,* dengan target responden 75 orang terhadap 11 pertanyaan kuesioner.

1. Pemeliharaan

Pada tahap pemeliharaan yaitu tahap dimana melakukan koreksi dari berbagai

*error* yang tidak ditemukan pada tahap-tahap sebelumnya sehingga dilakukan perbaikan, agar dapat menghasilkan sistem yang lebih baik dari sebelumnya. Dalam penggunaanya sistem search market tidak menutup kemungkinan akan terjadi *error* atau *bug,* untuk itu pemeliharaan atau pemeriksaan secara rutin dilakukan untuk melakukan *maintenance*. Penanganan *error* dan *bug* juga melalui pelaporan kesalahan sistem yang telah didapatkan oleh pengguna kepada *admin* melalui email “[irunwazed@gmail.com](mailto:irunwazed@gmail.com)”.

**3.4 Analisis Sistem**

Analisis sistem merupakan suatu tahapan yang bertujuan untuk mengetahui dan mengamati apa saja yang terlibat dalam suatu sistem. Pembahasan yang ada pada analisis sistem ini yaitu analisis kebutuhan fungsional meliputi perancangan sistem menggunakan bahasa pemodelan UML (*Unified Modeling Language*), perancangan tampilan *interface* serta analisis kebutuhan fungsoinal dan nonfungsional yang meliputi kebutuhan perangkat keras dan perangkat lunak yang akan digunakan.

**3.4.1 Analisis Kebutuhan Fungsional**

Analisis kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem agar sistem dapat berjalan sesuai dengan prosedur yang dibangun melalui perancangan sistem. Adapun kebutuhan fungsional dari sistem yang akan dibangun, yaitu;

1. Perancangan diagram sistem menggunakan bahasa pemodelan UML (*Unified Modeling Language*) yang meliputi pembuatan *flowchart* sistem, *flowchart* metode, *use case diagram*, *activity diagram, class diagram*  serta *sequence diagram.*
2. Menganalisis kebutuhan data yangterdiri dari data yang akan diambil dari *e-marketplace*, melakukan proses *web scraping* dari beberapa *e-marketplace* untuk menemukan beberapa data barang dan melakukan proses pencarian menggunakan algoritma *levenshtein distance* serta *output* dari aplikasi berupa hasil pencarian barang dari beberapa *e-marketplace*.

**3.4.2 Analisis Kebutuhan Non-fungsional**

Analisis kebutuhan non-fungsional adalah sebuah langkah dimana pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk membangun aplikasi yang akan dibangun. Analisis kebutuhan nonfungsional yang dilakukan dibagi dalam dua tahap, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak. Kebutuhan perangkat keras yaitu kebutuhan perangkat atau komponen yang dibutuhkan pada sistem dan perangkat lunak yaitu kebutuhan perangkat lunak untuk membantu agar komponen perangkat keras dapat berfungsi dan dapat dijalankan pada sistem.

**3.4.2.1 Kebutuhan Perangkat Keras**

Untuk menerapkan rancangan yang telah dijelaskan sebelumnya, dibutuhkan beberapa perangkat keras sebagai sarana untuk mengimplementasikan aplikasi yang dibangun. Berikut ini spesifikasi perangkat keras yang dibutuhkan.

**Tabel 4.1 Spesifikasi perangkat keras**

|  |  |  |
| --- | --- | --- |
| **No** | **Nama Perangkat** | **Spesifikasi** |
| 1. | *Notebook* | *Asus A455LD* |
| 2. | *Processor* | *Intel Core i5 2.10 GHz* |
| 3. | *Monitor* | *Monitor 14 inch* |
| 4. | *Memori* | *RAM 6 GB DDR 3L* |
| 5. | *Harddisk* | *500 GB HDD* |

**3.4.2.2 Kebutuhan Perangkat Lunak**

Perangkat lunak yang digunakan pada pembangunan aplikasi pencarian barang. Adapun rincian kebutuhan perangkat lunak dapat dilihat pada Tabel 4.2 berikut ini.

**Tabel 4.2 Spesifikasi perangkat lunak**

|  |  |  |
| --- | --- | --- |
| **No** | **Nama Perangkat** | **Spesifikasi** |
| 1. | *Operating System* | *Windows 10 Profesional 64 bit* |
| 2. | *Python* | *Python 3.6* |
| 3. | *Mysql* | *Mysqli* |
| 4. | *Apache* | *Php 5* |
| 5. | *Browser* | *Google Crome* |
| 6. | *Text Editor* | *Notepad++* |

**3.5 Analisis Perancangan Sistem**

Perancangan sistem yang akan dibangun terdiri atas perancangan *flowchar*t dan perancangan UML serta perancangan *user interface.*

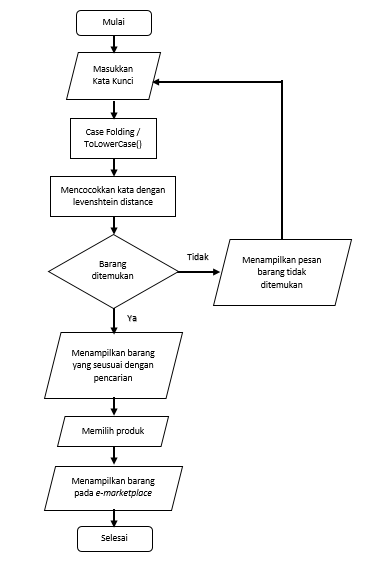
* + 1. **Perancangan *Flowchart***

*Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung. Perancangan *flowchart* yang akan dibangun terdiri atas *flowchart* sistem, *flowchart* algoritma *Levenshtein Distance*.

* + - 1. ***Flowchart* Sistem**

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* sistem pada aplikasi pencarian barang di berbagai *e-marketplace* yang ditunjukkan oleh Gambar 4.1. Adapun alur kerja *flowchart diagram* sistem adalah sebagai berikut:

1. *User* memasukkan kata yang ingin dicari.
2. Kata yang dimasukkan akan melalui proses *case folding/toLowerCase* untuk mengubah kata menjadi huruf kecil.
3. Kata yang dicari akan dicocokkan menggunakan algoritma *Levenshtein Distance*.
4. Menampilkan daftar produk yang berhubungan dengan kata kunci.
5. Jika produk tidak ditemukan, maka menampilkan pesan “Produk tidak ditemukan” dan *user* dapat melakukan pencarian ulang. Tapi jika barang ditemukan, maka akan menampilkan produk.
6. *User* memilih produk yang diinginkan.
7. Menampilkan halaman penjualan produk yang telah dipilih.

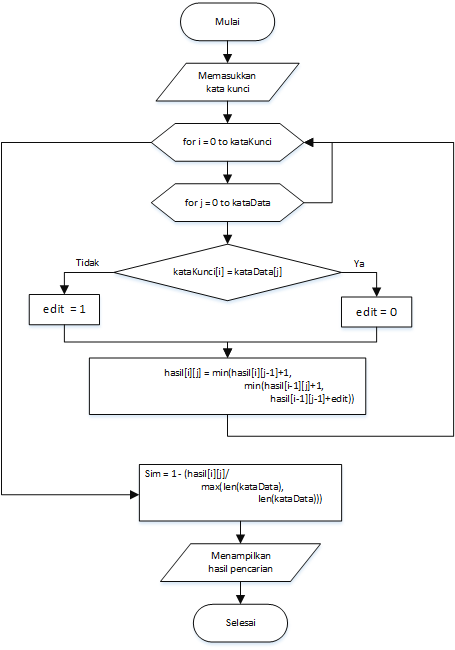


**Gambar 4.1 *Flowchart* Sistem**

* + - 1. ***Flowchart* algoritma *Levenshtein Distance***

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* algoritma *Levenshtein Distance* pada aplikasi pencarian barang di berbagai *e-marketplace* yang ditunjukkan oleh Gambar 4.2. Adapun alur kerja *flowchart diagram* algoritma *Levenshtein Distance* adalah sebagai berikut:

1. Pertama *user* memasukkan kata kunci yang akan dicari.
2. Sistem melakukan proses pencarian dengan syarat *for i =* panjang kata kunci dimana *i=0*.
3. Sistem melakukan proses pencarian dengan syarat *for j =* panjang kata data dimana *j=0*.
4. Kemudian mengecek huruf pada kata kunci dan kata data. Bila huruf tersebut berbeda, maka memberikan tambahan nilai 1 kata sebelumnya.
5. Selanjutnya mencari nilai minimal dari nilai kata kunci, kata data dan kata sebelumnya.
6. Setelah itu, huruf selanjutnya akan kembali diproses dengan cara yang sama. Apabila huruf yang akan diproses sudah tidak ada lagi, maka system akan menampilkan hasil pencarian.

 **Gambar 4.2 *Flowchart* Algoritma *Levenshtein Distance***

Berdasarkan gambar 4.3. pencarian nilai Levenshtein Distance dapat dilakukan dengan memasukkan kedalam tabel kemudian melakukan perhitungan sebagai berikut:

**Tabel 4.3 Inisialisasi Awal *Levenshtein Distance***

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

B­erdasarkan tabel 4.3, inisialisasi awal dilakukan dengan menyusun kata pertama secara vertikal dan kata kedua secara horizontal, lalu diberikan nilai secara berurutan pada setiap katanya.

**Tabel 4.4 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 4.4 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 4.5 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 4.5 terbapat lanjutan baris yang dilakukan sama dengan cara sebelumnya. Dengan mencari nilai minimum dari beberapa nilai kotak.

**Tabel 4.6 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 4.6 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* kedalam *similarity* sebagai berikut:

**(4.1)**

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.

* + 1. ***Unified Modeling Language* (UML)**

Aplikasi dibangun dengan menggunakan *Unified Modeling Language* (UML). UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram yang terdiri dari *Use Case Diagram, Activity Diagram, Class Diagram* dan *Sequence Diagram.*

* + - 1. ***Use Case Diagram***

*Use Case Diagram* adalah sebuah diagram yang dapat merepresentasikan interaksi yang terjadi antara user dengan sistem. *Use Case Diagram* ini mendeskripsikan siapa saja yang menggunakan sistem dan bagaimana cara mereka berinteraksi dengan sistem. *Use Case Diagram* dari sistem yang akan dibangun dapat ditunjukkan pada Gambar 4.3.



**Gambar 4.3 *Use Case Diagram* Aplikasi**

**Tabel 4.7 Keterangan *Use Case Diagram***

|  |  |
| --- | --- |
| **Aktor** | **Sistem** |
| *User* memilih menu beranda | Sistem akan menampilkan menu beranda. |
| *User* memilih pencarian | Sistem akan menyiapkan *form* pencarian*.* |
| *User* memilih menu petunjuk | Sistem akan menampilkan menu petunjuk bantuan penggunaan aplikasi. |
| *User* memilih menu tentang | Sistem akan menampilkan menu tentang aplikasi. |
| *User* memilih menu login | Sistem akan menampilkan *form* *login*. |
| *User* memilih menu cari data | Sistem akan menampilkan form beserta pilihan *website* yang akan diambil datanya. |
| *User* memilih menu lihat data | Sistem akan menampilkan data-data yang tersedia pada *website*. |
| *User* memilih menu halaman admin | Sistem akan menampilkan halaman awal admin |

* + - 1. ***Activity Diagram***

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

1. ***Activity Diagram* Beranda**

Gambar 4.4 merupakan *activity diagram* yang menunjukkan aktivitas *user* ketika memilih menu beranda, lalu sistem akan menampilkan menu beranda yang berisikan tampilan menu beranda.



**Gambar 4.4 *Activity Diagram* Beranda**

1. ***Activity Diagram* Pencarian**

Gambar 4.5 tersebut merupakan *activity diagram* yang menunjukkan aktivitas *user* ketika telah memilih menu cariproduk*,* lalu sistem akan menampilkan *form input* kata, *user* kemudian memasukkan kata yang dicari lalu menekan tombol cari dan sistem akan menampilkan daftar produk yang dicari.

**Gambar 4.5 *Activity Diagram* Pencarian**

1. ***Activity Diagram* Petunjuk**

Gambar 4.6 merupakan gambar *activity diagram* yang menunjukkan aktivitas *user* memilih menu petunjuk, maka sistem akan menampilkan menu petunjuk. Setelah itu, menu petunjuk tersebut akan menampilkan cara penggunaan sistem.

 **Gambar 4.6 *Activity Diagram* Petunjuk**

1. ***Activity Diagram* Tentang**

Gambar 4.7 merupakan gambar *activity diagram* yang menunjukkan aktivitas *user* memilih menu tentang, maka sistem akan menampilkan menu tentang. Setelah itu, menu tersebut akan menampilkan penjelasan sistem.

 **Gambar 4.7 *Activity Diagram* Tentang**

1. ***Activity Diagram* Login**

Gambar 4.8 merupakan gambar *activity diagram* yang menunjukkan aktivitas adminmemilih menu *login*, maka sistem akan menampilkan *form login*. Setelah itu, admin mengisi *form login* untuk masuk ke halaman admin.



**Gambar 4.8 *Activity Diagram Login***

1. ***Activity Diagram* Cari Data**

Gambar 4.9 merupakan gambar *activity diagram* yang menunjukkan aktivitas admin yang telah masuk ke halaman admin untuk memilih menu cari data, maka sistem akan menampilkan *form* cari data. Setelah itu, admin mengisi *form* sesuai kebutuhan.



**Gambar 4.9 *Activity Diagram* Cari Data**

1. ***Activity Diagram* Lihat Data**

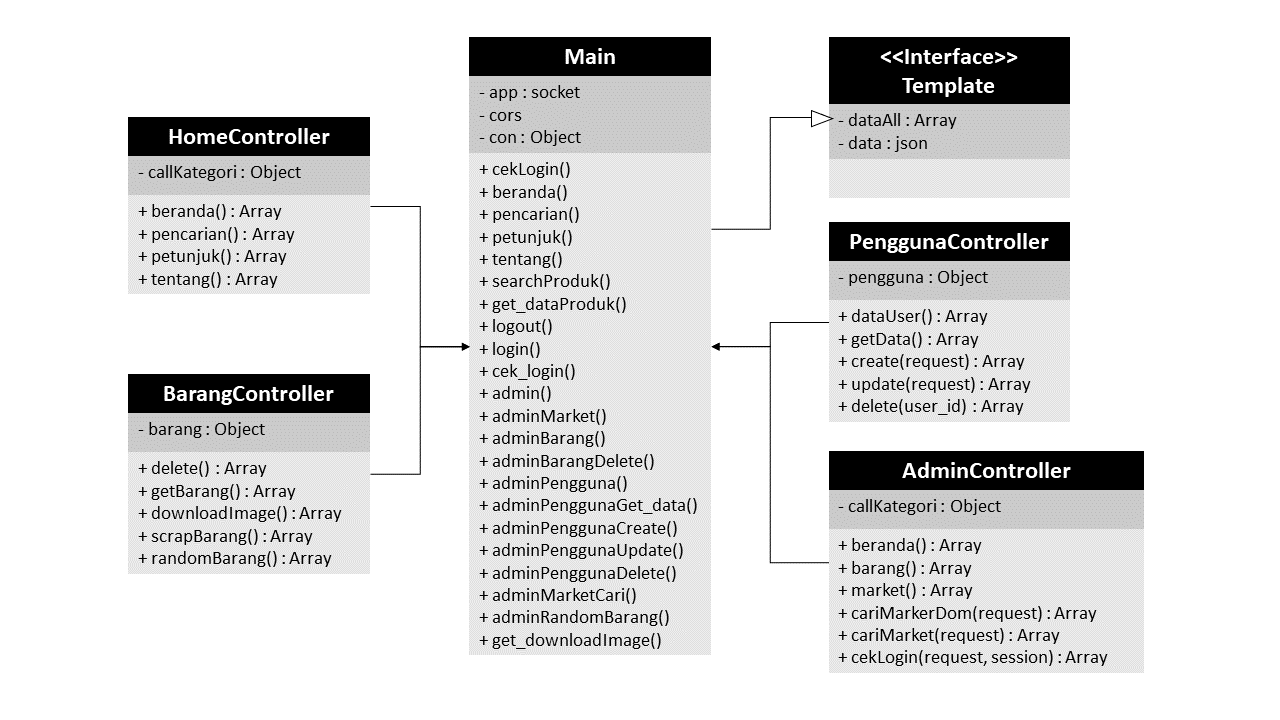
Gambar 4.10 merupakan gambar *activity diagram* yang menunjukkan aktivitas admin yang telah masuk ke halaman admin untuk memilih menu lihat data, maka sistem akan menampilkan daftar data yang ada.



**Gambar 4.10 *Activity Diagram* Lihat Data**

* + - 1. ***Class Diagram***

*Class diagram* merupakan diagram yang selalu ada dipemodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Berikut ini adalah *class diagram* sistem:

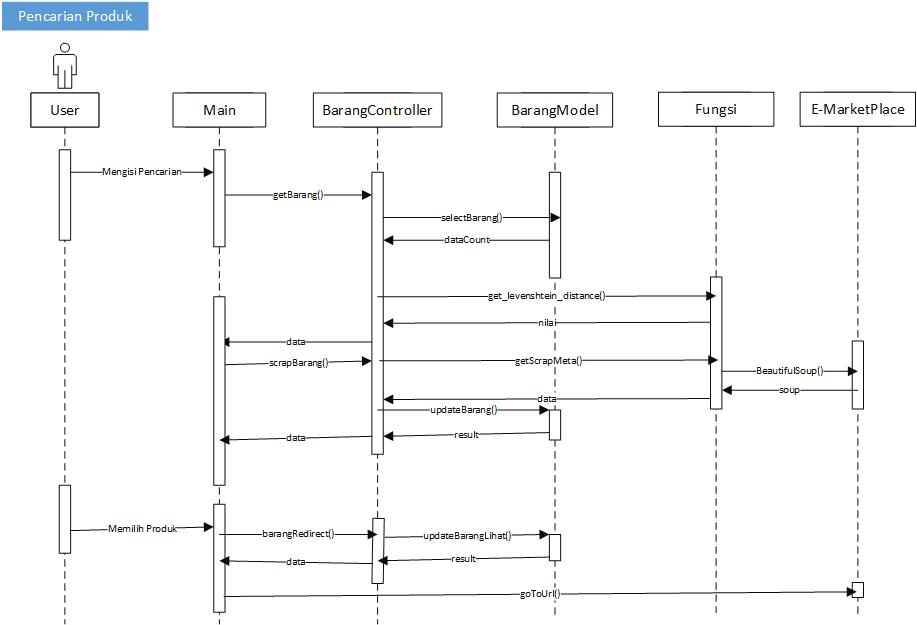
****

**Gambar 4.11 *Class Diagram* Aplikasi Pencarian**

* + - 1. ***Sequence Diagram***

*Sequence Diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang digambarkan terhadap waktu. Berikut ini adalah *Sequence Diagram* yang akan menggambarkan interkasi antar objek dan sistem.

1. ***Sequence Diagram* Pencarian Produk**

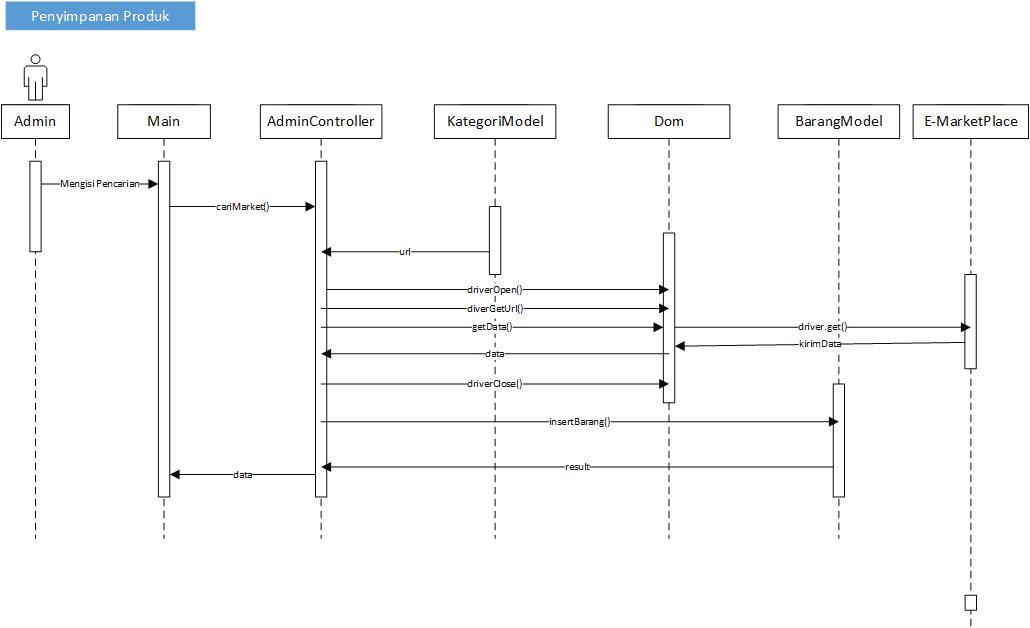
******

**Gambar 4.12 *Sequence Diagram* Pencarian Produk**

Gambar 4.12 adalah *sequence* *diagram* yang menunjukkan hubungan dari sistem terhadap *user*, dimana sistem akan menampilkan *form* pencarian, lalu *user* memasukkan kata kunci untuk mencari barang yang diinginkan.

1. ***Sequence Diagram* Penyimpanan Produk**

Gambar 4.13 adalah diagram *sequence* yang menunjukkan hubungan dari sistem terhadap *admin*, dimana *admin* memilih menu cari data. Kemudian *admin* memilih *website* yang akan secara otomatis mencari data pada *website* tersebut.

******

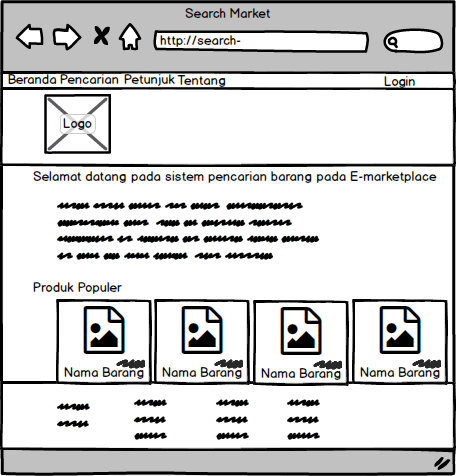
**Gambar 4.13 *Sequence Diagram* Penyimpanan Produk**

* 1. **Perancangan Antarmuka (Interface)**

Perancangan *user interface* adalah tahapan pembuatan antarmuka yang akan digunakan pada pembangunan aplikasi pencarian barang di berbagai *e-marketplace* yang dibagi menjadi sembilan bagian yaitu perancangan menu beranda, *form* pencarian, menu petunjuk, menu tentang, *form* login, menu halaman admin, *form* cari data dan *form* lihat data.

* + 1. **Rancangan Menu Beranda**

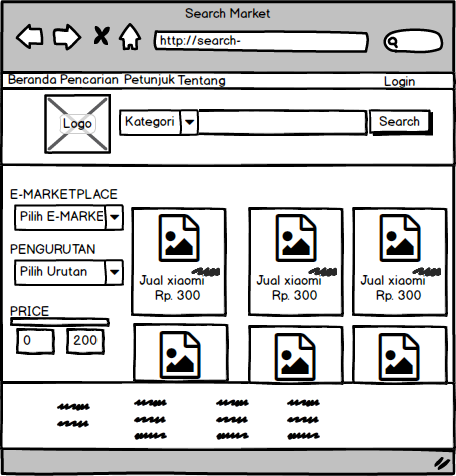
Menu beranda adalah tampilan pertama yang akan digunakan pada *website*. Pada menu beranda berisi logo aplikasi.



**Gambar 4.14 Rancangan Menu Beranda**

* + 1. **Rancangan *Form* Pencarian**

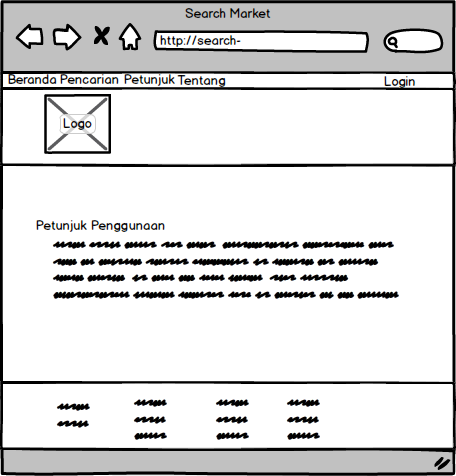
Pada *form* pencarian *user* memasukkan kata kunci untuk mencari produk yang sesuai dengan kata kunci. Kemudian sistem akan menampilkan daftar produk yag terkait.

****

**Gambar 4.15 Rancangan *Form* Pencarian**

* + 1. **Rancangan Menu Petunjuk**

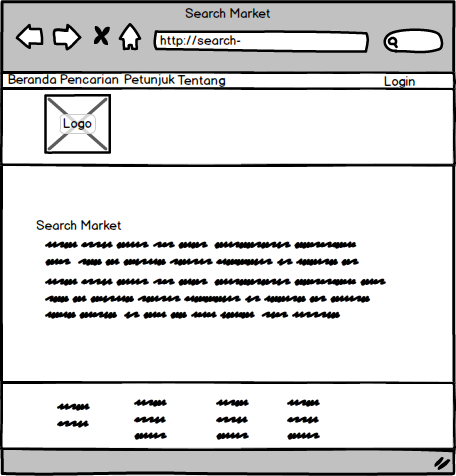
Pada saat *user* memilih menu petunjuk, sistem akan menampilkan petunjuk penggunaan sistem. Disini akan dijelaskan cara-cara penggunaan aplikasi untuk mencari produk.

****

**Gambar 4.16 Rancangan Menu Petunjuk**

* + 1. **Rancangan Menu Tentang**

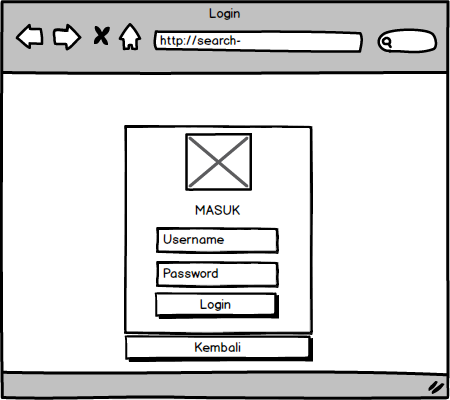
Pada saat *user* memilih menu tentang, sistem akan menampilkan penjelasan mengenai deskripsi sistem. Sehingga *user* dapat mengetahui manfaat dari sistem ini.



**Gambar 4.17 Rancangan MenuTentang**

* + 1. **Rancangan *Form* *Login***

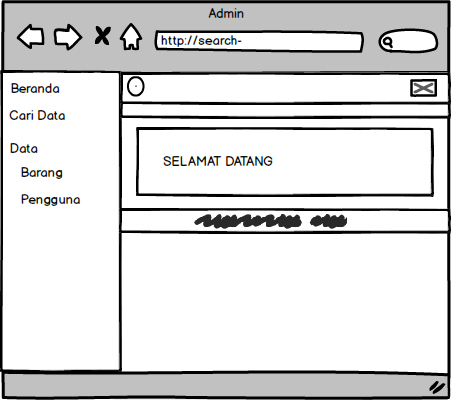
Pada saat *user* memilih menu login, sistem akan menampilkan *form* *login* sebagai *admin*. Dengan cara memasukkan *username* dan *password* dengan benar.

****

**Gambar 4.18 Rancangan *Form Login***

* + 1. **Rancangan Menu Halaman Admin**

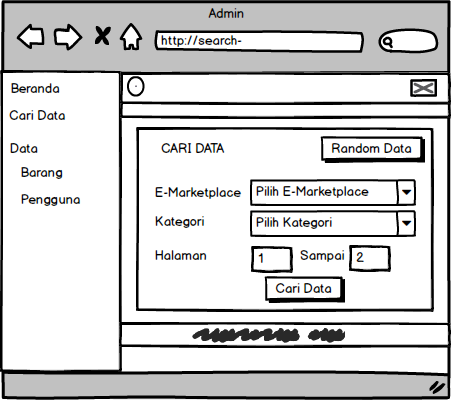
Pada saat *user* memilih menu Halaman Admin, sistem akan menampilkan tampilan awal setelah melakukan *login*. Menu ini berisi ucapan selamat dating.

****

**Gambar 4.19 Rancangan MenuHalaman Admin**

* + 1. **Rancangan Menu Cari Data**

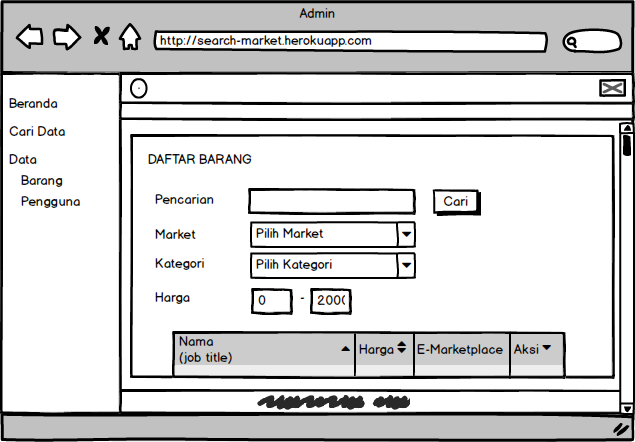
Pada saat *user* memilih menu Cari Data, sistem akan menampilkan pilihan website yang akan diambil datanya dengan menggunakan teknik *web scraping*.

****

**Gambar 4.20 Rancangan MenuCari Data**

* + 1. **Rancangan Menu Barang**

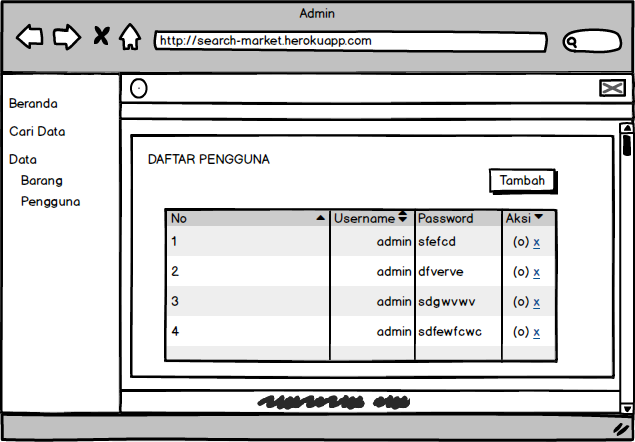
Pada saat *user* memilih menu Barang, sistem akan menampilkan daftar produk yang ada pada *database*.

****

**Gambar 4.21 Rancangan MenuBarang**

* + 1. **Rancangan Menu Pengguna**

Padas sat *user* memilih menu Pengguna, sistem akan menampilkan daftar pengguna yang berguna sebagai login *user*.



**Gambar 4.22 Rancangan MenuPengguna**

* 1. **Metode Pengujian**

Pengujian merupakan metode yang dilakukan untuk menjelaskan mengenai pengoperasian perangkat lunak yang terdiri dari perangkat pengujian, metode pengujian dan pelaksanaan pengujian.

Pengujian program ini menggunakan metode *black box*. Pengujian *black box* merupakan pengujian program berdasarkan fungsi dari program. Tujuan dari metode *black box* ini adalah untuk menemukan kesalahan fungsi pada program. Pengujian dengan metode ini dilakukan dengan cara memberikan sejumlah *input* pada program aplikasi yang kemudian diproses sesuai dengan kebutuhan fungsionalnya untuk melihat apakah program aplikasi menghasilkan keluaran yang diinginkan dan sesuai dengan fungsi dari program tersebut. Apabila dari masukan yang diberikan proses menghasilkan keluaran yang sesuai dengan kebutuhan fungsionalnya, maka program aplikasi yang bersangkutan telah benar, tetapi jika keluaran yang dihasilkan tidak sesuai dengan kebutuhan fungsionalnya, maka masih terdapat kesalahan pada program aplikasi.

Pengujian dilakukan dengan mencoba semua kemungkinan yang terjadi dan pengujian dilakukan berulang-ulang. Jika dalam pengujian ditemukan kesalahan, maka akan dilakukan penelusuran dan perbaikan untuk memperbaiki kesalahan yang terjadi. Jika telah selesai melakukan perbaikan, maka akan dilakukan pengujian kembali. Pengujian dan perbaikan dilakukan secara terus menerus hingga diperoleh hasil yang terbaik.

* 1. **Rencana Pengujian**

Pengujian perangkat lunak berikut menggunakan data uji berdasarkan dari masing-masing data. Rencana selengkapnya dapat dilihat pada tabel berikut:

**Tabel 4.8 Rencana Pengujian Black Box**

|  |  |  |
| --- | --- | --- |
| **Kelas Uji** | **Detail Uji** | **Jenis Pengujian** |
| Pengujian Menu | Beranda | *Black Box* |
| Pencarian | *Black Box* |
| Petunjuk | *Black Box* |
| Tentang | *Black Box* |
| Login | *Black Box* |
| Pengujian Menu Admin | Halaman Admin | *Black Box* |
| Lihat Data | *Black Box* |
| Cari Data | *Black Box* |
| Pengujian Algoritma *Levenshtein Distance* | Pencarian data yang ada di *database* | *Black Box* |
| Pengujian Teknik *Web Scraping* | Pencarian data pada website luar. | *Black Box* |

Setelah melakukan pengujian *black box*, sistem akan dilakukan pengujian kembali kepada para pengguna agar mengetahui pendapat pengguna mengenai sistem yang dikembangkan. Pengguna melakukan pengisian kuesioner untuk memberikan pendapatnya terhadap sistem. Kuesioner terdiri dari 11 pertanyaan dengan 5 opsi yang dapat dipilih sesuai tanggapan oleh pengguna. Setelah mendapatkan respon dari pengguna, data tersebut akan dihitung tinggat kesuksesan sistem sesuai pada tabel 4.9.

**Tabel 4.9 Rencana Kuesioner**

|  |  |  |  |
| --- | --- | --- | --- |
| **Optional** | **Bobot** | **Interval Interprestasi skor** | **Jenis Pengujian** |
| Sangat Setuju | 5 | 80% – 100% | Kuesioner |
| Setuju | 4 | 60% – 79,99% | Kuesioner |
| Netral | 3 | 40% – 59,99% | Kuesioner |
| Tidak Setuju | 2 | 20% – 39,99% | Kuesioner |
| Sangat Tidak Setuju | 1 | 0% – 19,99% | Kuesioner |
| **Rumus memperoleh skor** |  | | Kuesioner |

**DAFTAR PUSTAKA**

Ariyani, N. H., Sutardi, & Ramadhan, R. (2016). Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance. *SemanTIK*, *Vol 2*(1), 279–286.

Azizah, U. N. (2013). Perbandingan Detektor Tepi Prewit Dan Detektor Tepi Laplacian Berdasarkan Kompleksitas Waktu Dan Citra Hasil. UNIVERSITAS PENDIDIKAN INDONESIA.

Indrajit, R. E. (2012, November 16). Evolusi E-Marketplace. *E-Artikel Sistem Dan Teknologi Informasi*, *4*(C), 1–4.

Nathasya. (2018). ONLINE SHOP MARKETPLACE DAN E-COMMERCE SEBAGAI PERKEMBANGAN TEKNOLOGI MODERN. *PT Dewaweb Indonesia*, pp. 1–4.

Panjaitan, P. M. (2018). *Sistem Monitoring Cuaca dan Identifikasi Keadaan Cuaca Menggunakan Teknik Web Scraping*. UNIVERSITAS SUMATERA UTARA.

Pratama, B. P., & Pamungkas, S. A. (2016). Analisis Kinerja Algoritma Levenshtein Distance dalam Mendeteksi Kemiripan Dokumen Teks. *Jurnal “LOG!K@,”* *6*(2), 131–143.

Purba, A. H., & Situmorang, Z. (2017). Analisis Perbandingan Algoritma Rabin-Karp Dan Levenshtein Distance Dalam Menghitung Kemiripan Teks. *Jurnal Teknik Informatika Unika St. Thomas (JTIUST)*, *02*, 24–32.

Ryanda, R. (2015). *PERANCANGAN APLIKASI MOBILE “KIOSKU.COM” DENGAN WEB SCRAPPING PADA WEBSITE OLX.CO.ID, BERNIAGA.COM, DAN BUKALAPAK.COM BERBASIS ANDROID*. UNIVERSITAS TELKOM.

Sembiring, J. P. (2013). Perancangan Aplikasi Kamus Bahasa Indonesia - Karo Online Berbasis Web Dengan Metode Sequential Search. *Pelita Informatika Budi Darma*, *4*(2), 28–33.

Shalahudin. (2011). *Modul Pembelajaran Rekayasa Perangkat Lunak : Terstruktur dan Beorientasi Objek*. Bandung: Modula.

Soesanto, A. L. (2010). *PEMBANGUNAN SITUS E-MARKETPLACE UNTUK PERSEWAAN VILLA MENGGUNAKAN SMARTY DAN JQUERY*. UNIVERSITAS ATMA JAYA YOGYAKARTA.

Tim APJII. (2018). Potret Zaman Now Pengguna dan Perilaku Internet Indonesia. *BUletin APJII*, 1–7.

Utomo, M. S. (2013). Web Scraping pada Situs Wikipedia menggunakan Metode Ekspresi Regular. *Jurnal Teknologi Informasi DINAMIK*, *18*(2), 153–160.

Vladimir, Z. (2015). Electronic Commerce: Structures and Issues. *International Journal of Electronic Commerce*, *1*(1), 3–23.

Winardi, P., Abdillah, M. I., Hermanto, D., & Widiyanto, E. P. (2018). *Analisis Perbandingan Performa Web Server Berbasis PHP dan Web Server Berbasis Python Twisted*. (x).

Wiradinata, T. (2013). *Factors Influencing Nascent Entrepreneurs in an e-Marketplace*. *3*(4).

Witono, T., & Hendrayana, F. (2011). *Pembuatan Aplikasi Web Jual Beli dan Lelang Online Pendahuluan Tinjauan Pustaka*. *6*(65), 101–111.