**BAB V**

**IMPLEMENTASI DAN PENGUJIAN SISTEM**

* 1. **Implementasi Sistem**

Implementasi merupakan tahap di mana sistem telah siap untuk digunakan. Hasil analisis dan perancangan sistem yang telah dilakukan sebelumnya diimplementasikan dalam bentuk aplikasi berupa sebuah *web* pencarian barang di berbagai *e-marketplace* mengunakan teknik *web scraping* dan algoritma *levenstein distance* yang membantu pengguna untuk memudahkan dalam menemukan dan membandingkan barang yang diinginkan di berbagai *e-marketplace* dengan bahasa pemrograman phytonsebagai *server*. Untuk pembuatan tampilannya menggunakan HTML*,* CSS dan Javascript.

* + 1. **Kebutuhan Sistem**

Untuk menjalankan sistem implementasi *Levenshtein Distance* pada Aplikasi Pencarian Barang di Berbagai *E-Marketplace* Menggunakan Teknik *Web Scraping* diperlukan perangkat lunak sebagai berikut:

1. *MySQL* sebagai basis data sistem.
2. *Web Browser* untuk menampilkan sistem.

Sistem pencarian barang di berbagai *e-marketplace* telah diuji dan dapat beroperasi pada laptop dengan spefisikasi sebagai berikut:

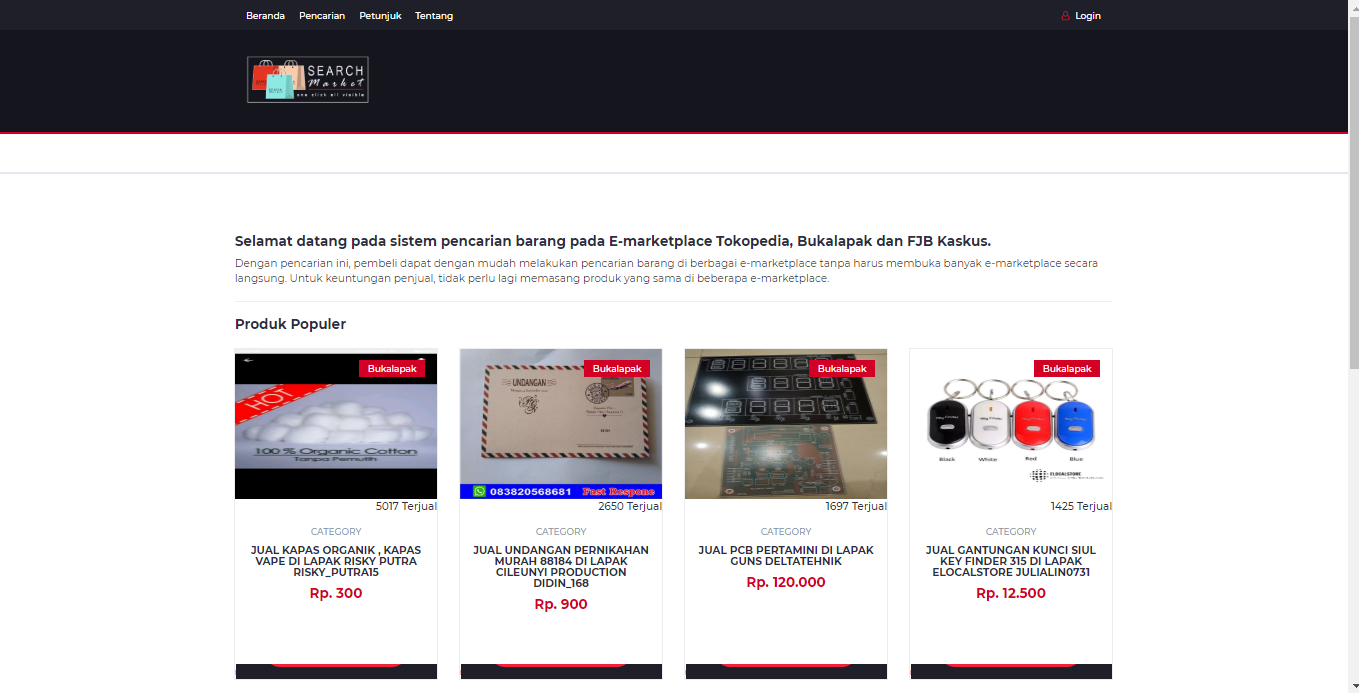
1. Sistem Operasi Windows 10 *Professional 64-bit*.
2. *Processor Intel*® *Core*TM i5-5200 *CPU* 2.20 *GHz*.
3. HDD SATA 512 GB
4. RAM 6.00 GB DDR 3L.
   * 1. **Standar Operasional Prosedur (SOP) Sistem**

Untuk mengoptimalkan penggunaan sistem implementasi *Levenshtein Distance* pada Aplikasi Pencarian Barang di Berbagai *E-Marketplace* terdapat beberapa hal yang perlu diperhatikan yaitu:

1. Sistem memiliki fitur mengambil data dari beberapa *e-marketplace* menggunakan teknik *web scraping*.
2. Sistem ini hanya melakukan pencarian barang dari *E-marketplace* yang terdaftar yaitu Bukalapak,Tokopedia dan FJB kaskus.
3. Sistem ini memiliki fitur pencarian barang menggunakan *filter* kategori barang, *filter* kisaran harga, *filter* urutan dan *filter* dari *e-marketplace* yang diinginkan.
   * 1. **Implementasi Tampilan Antarmuka Sistem**

*Interface* diambil dari *website* yang telah dipublish ke publik agar dapat diakses secara global dengan *URL* : **https://search-market.herokuapp.com/**

1. **Menu Beranda (*User*)**

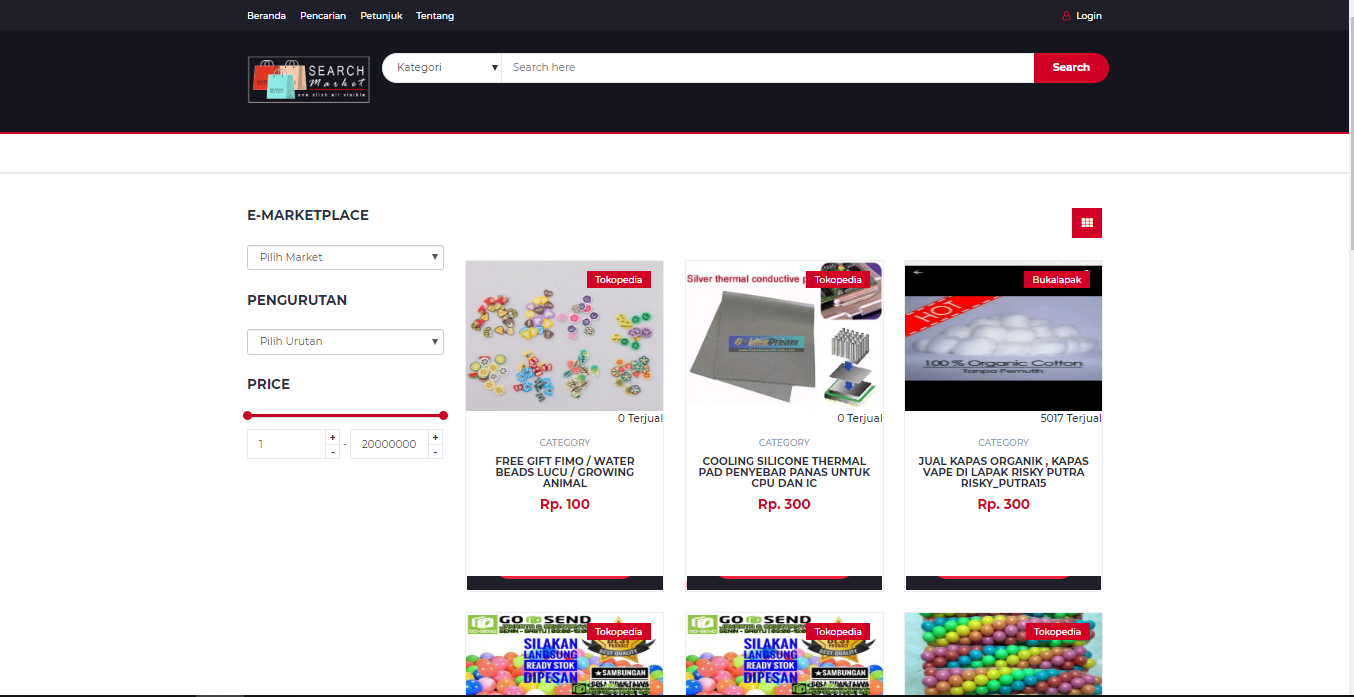


**Gambar 5.1 Tampilan Menu Beranda (*User)***

Pada menu antarmuka aplikasi untuk *user*, terdapat beberapa menu, yaitu beranda, pencarian, petunjuk, tentang dan login.

1. **Menu Pencarian *User***

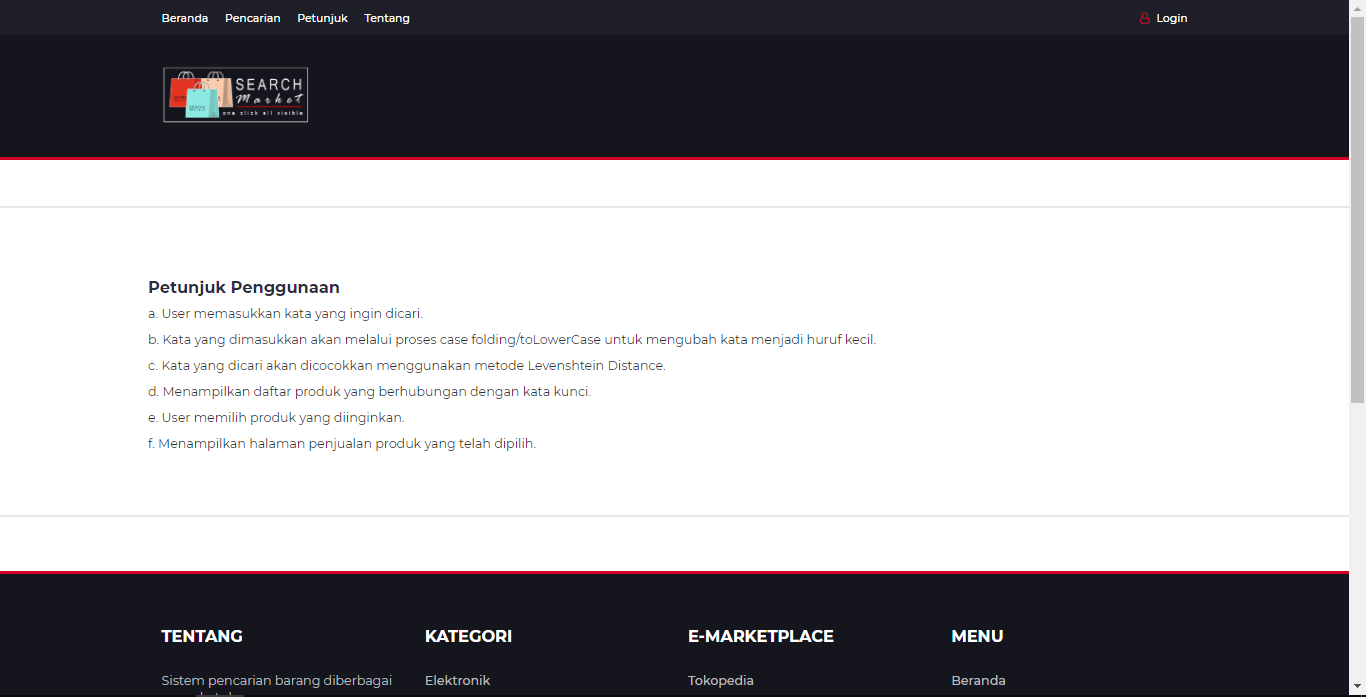
Pada menu pencarian *user*, terdapat kolom pencarian yang berfungsi untuk membantu proses pencarian barang dan juga terdaftar *filter-filter* yang membantu proses pencarian barang berdasarkan kategori barang,kisaran rentang harga barang, pengurutan barang dan *e-marketplace* dari barang yang diinginkan.



**Gambar 5.2 Tampilan *Form* Pencarian (*User*)**

1. **Menu Petunjuk (*User*)**

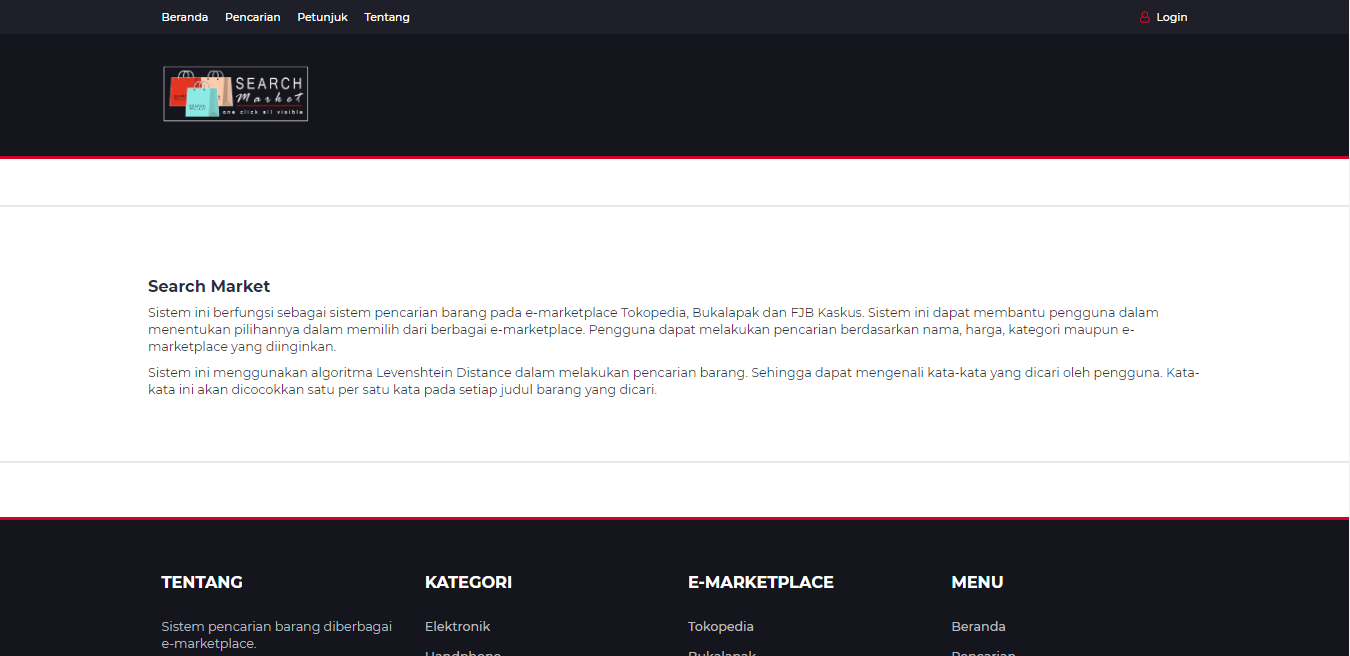
Pada menu petunjuk, ditampilkan petunjuk-petunjuk penggunaan aplikasi pencarian barang di berbagai *e-marketplace*.



**Gambar 5.3 Tampilan Menu Petunjuk (*User*)**

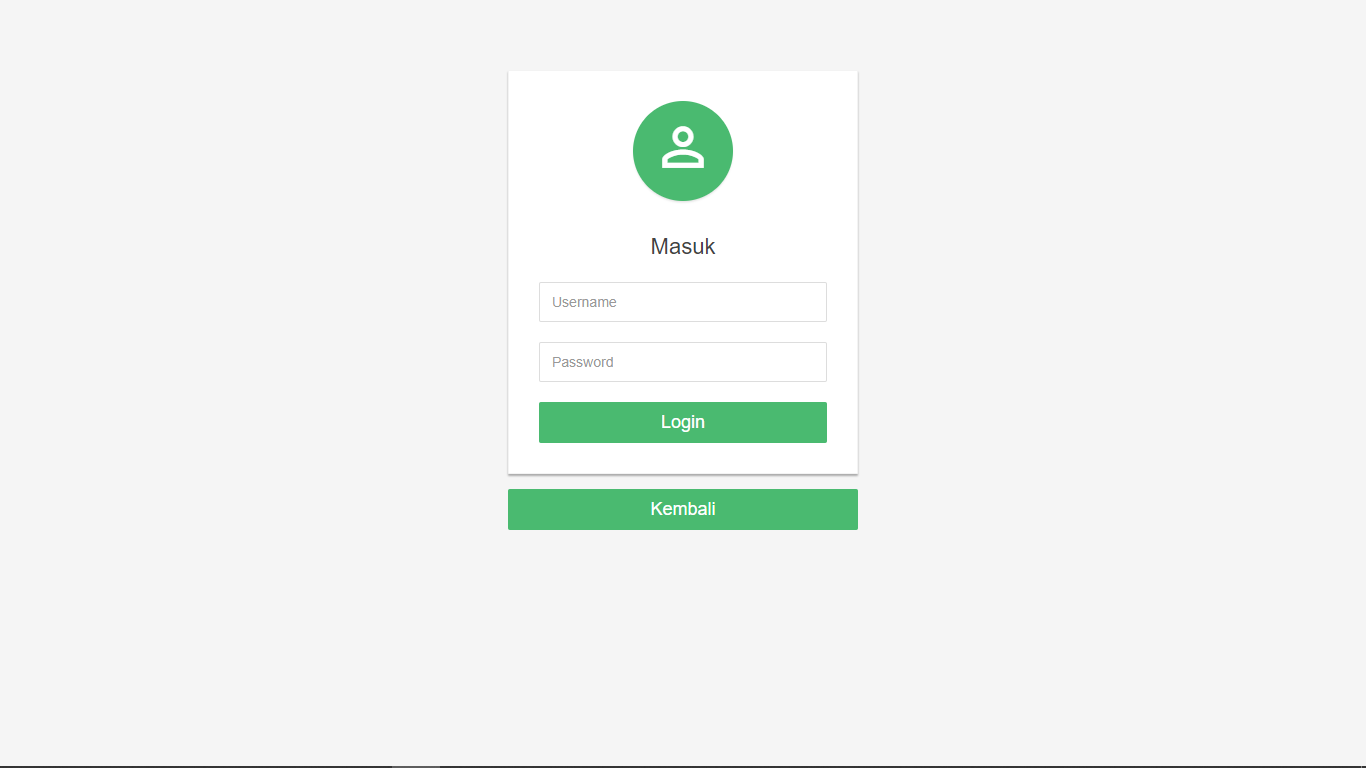
1. **Menu Tentang (*User*)**

Menu tentang, menampilkan deskripsi-deskripsi dari aplikasi seperti fungsi aplikasi, *e-marketplace* yang terdaftar dan algoritma pencarian yang digunakan.



**Gambar 5.4 Tampilan Menu Tentang (*User*)**

1. **Menu *Login* *Admin***

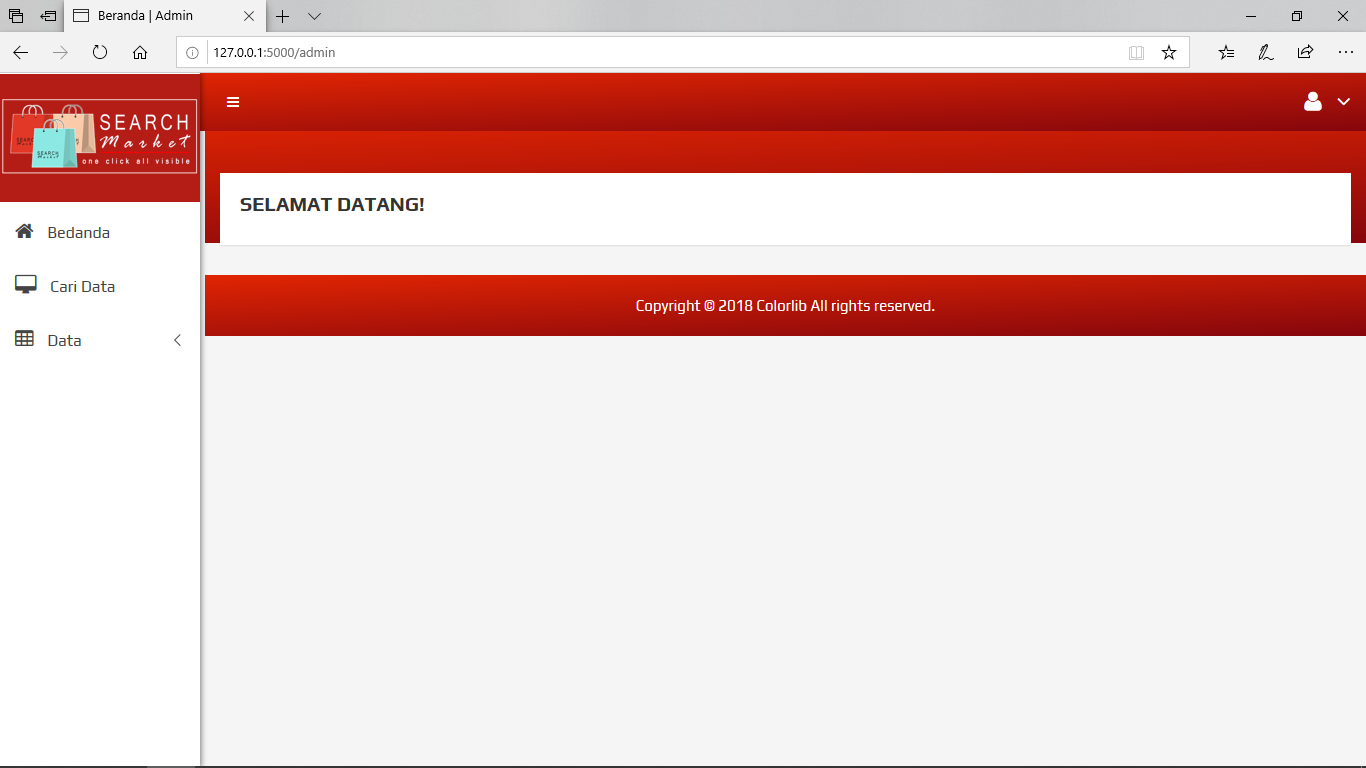


**Gambar 5.5 Tampilan Menu *Login* *Admin***

Sebelum dapat mengakses halaman beranda *admin*, pengguna diwajibkan mengisi *form* *login*. Terdiri dari dua kolom pengisian untuk mengisi *username* dan *password* pengguna. Terdapat dua tombol lainnya, tombol login berfungsi untuk masuk ke halaman amin dan tombol kembali berfungi untuk kembali ke halaman sebelumnya. Jika pengguna memasukkan data *login* yang salah, maka akan kembali lagi ke halam *login*.

1. **Menu Tampilan Beranda (*Admin*)**

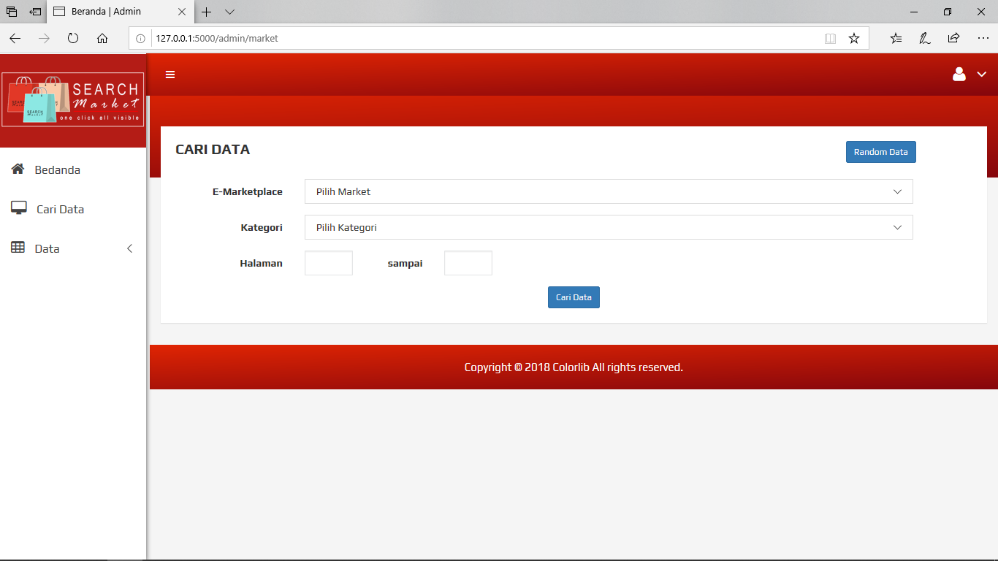
Setelah pengguna melakukan *login*, pengguna akan diarahkan ke halaman beranda *admin*. Pada menu antarmuka aplikasi untuk *admin*, terdapat beberapa menu, yaitu menu beranda, menu cari data dan menu data yang terdiri dari dua bagian yaitu data barang dan data pengguna.



**Gambar 5.6 Tampilan Menu Beranda (*Admin*)**

1. **Menu Cari Data (*Admin*)**

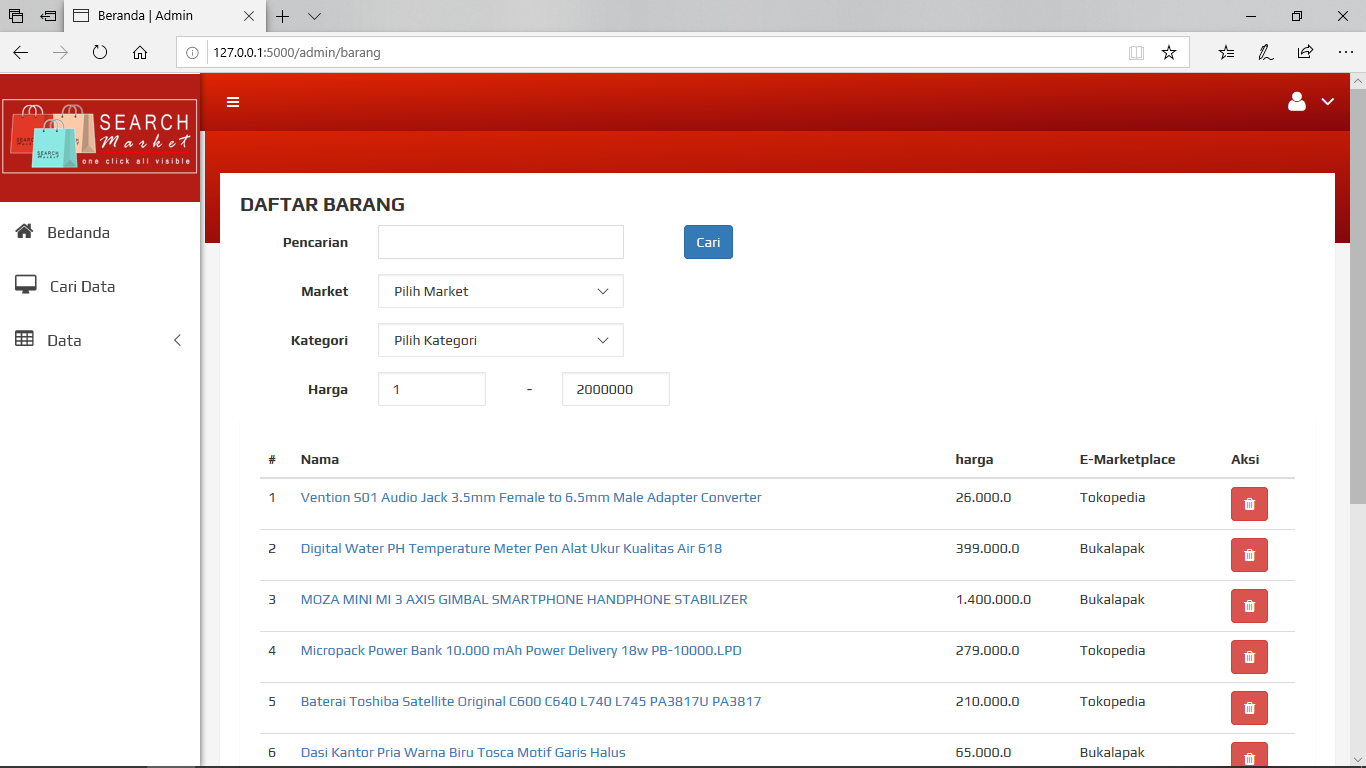
Pada menu cari data berfungsi untuk mencari barang mengunakan teknik web scraping dari *e-marketplace* yang dipilih untuk menambahkannya ke dalam basis data sistem.



**Gambar 5.7 Tampilan Menu Cari Data (*Admin*)**

1. **Menu Data Barang (*Admin*)**

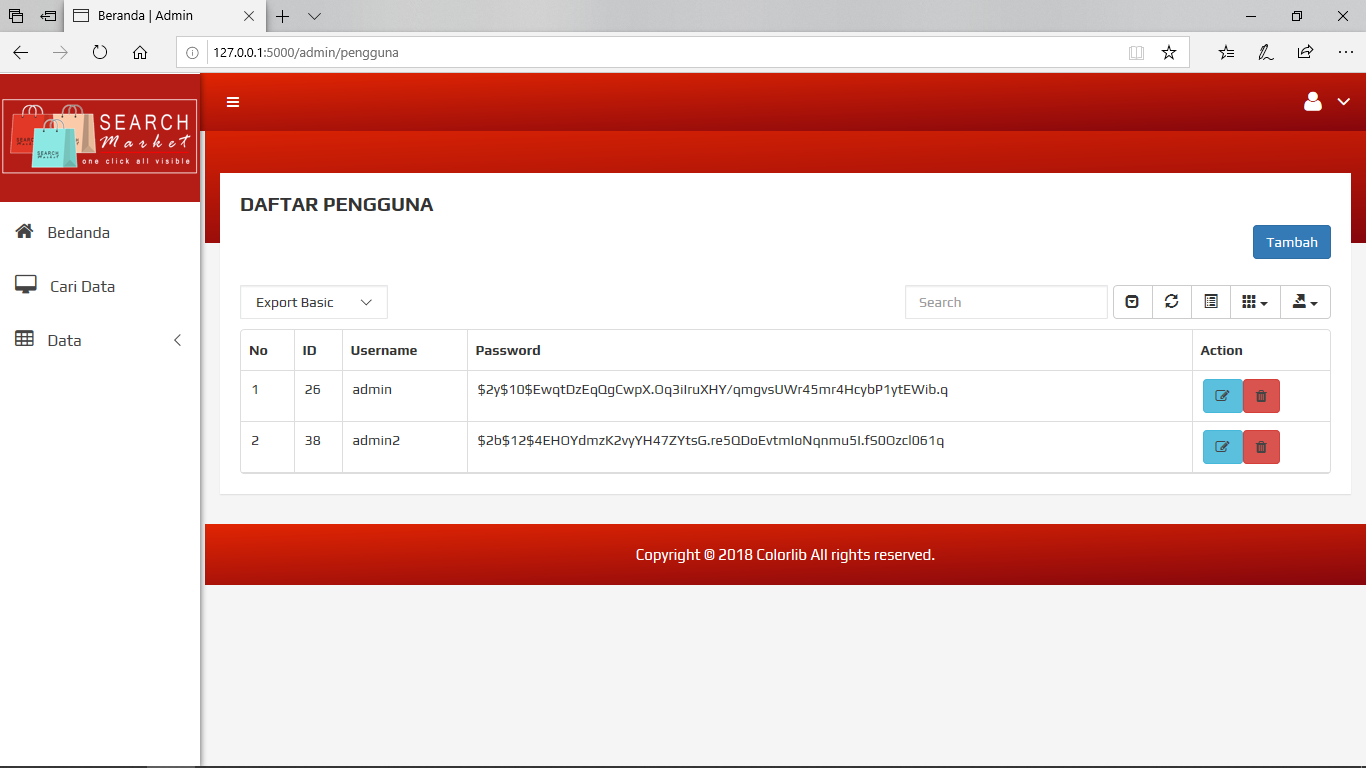
Pada menu data admin akan ditampilkan data-data barang yang berhasil ditambahkan menggunakan teknik *web scraping* ke basis data sistem. Disini admin dapat melakukan pencarian barang dan menghapus data barang.

****

**Gambar 5.8 Tampilan Menu Data Barang (*Admin*)**

1. **Menu Data Pengguna (*Admin*)**

Pada menu data pengguna akan ditampilkan data-data pengguna admin yang telah terdaftar. Disini admin dapat melakukan tambah, ubah dan hapus data admin.

****

**Gambar 5.9 Tampilan Data Pengguna (*Admin*)**

**5.2 Implementasi Program**

**5.2.1 Implementasi Program *Web Scraping***

Gambar 5.10 merupakan *source code* program *Web Scraping*. *Source code* ini digunakan dalam proses ekstraksi data dari *e-marketplace* yang akan dimasukkan ke *database* sistem*.*

|  |
| --- |
| def getScrapMeta(self, \_url, nilai = 0):  import requests  import re  from bs4 import BeautifulSoup    start = time.time()  webpageurl = \_url  hdr = {'User-Agent': 'Mozilla/5.0'}  data = requests.get(\_url,headers=hdr)  soup = BeautifulSoup(data.text,'html.parser')  #soup = BeautifulSoup(data.text,'lxml')  nama = ''  harga = ''  info = ''  img = ''  status = False  try:  nama = soup.find("meta",  attrs={"name":"twitter:title"})["content"]  harga = soup.find("meta",  attrs={"name":"twitter:data1"})["content"]  info = soup.find("meta",  attrs={"name":"twitter:description"})["content"]  img = soup.find("meta",  attrs={"name":"twitter:image"})["content"]  harga = re.findall("\d+", harga)  harga = ''.join(harga)  status = True    except Exception as e:  print("Excepton found", format(e))  print(\_url)  end = time.time()  delay = end - start  kirim = {  'status': status,  'nama': nama,  'harga': harga,  'img': img,  'info': info,  'nilai': nilai,  'url': \_url,  'delay': delay,  }  return kirim |

**Gambar 5.10 Tampilan *Source Code* Program *Web Scraping***

**5.2.2 Implementasi Algoritma *Levenstein Distance***

Gambar 5.11 merupakan *source code* Algoritma *Levenshtein Distance*, dimana *source code* ini digunakan dalam pencarian data barangpada sistem aplikasi.

|  |
| --- |
| def get\_levenshtein\_distance(self, word1, word2):  word2 = word2.lower()  word1 = word1.lower()  matrix = [[0 for x in range(len(word2) + 1)] for x in range(len(word1) + 1)]  for x in range(len(word1) + 1):  matrix[x][0] = x  for y in range(len(word2) + 1):  matrix[0][y] = y  for x in range(1, len(word1) + 1):  for y in range(1, len(word2) + 1):  if word1[x - 1] == word2[y - 1]:  matrix[x][y] = min(  matrix[x - 1][y] + 1,  matrix[x - 1][y - 1],  matrix[x][y - 1] + 1  )  else:  matrix[x][y] = min(  matrix[x - 1][y] + 1,  matrix[x - 1][y - 1] + 1,  matrix[x][y - 1] + 1  )  nilai = matrix[len(word1)][len(word2)]  hasil = 1 – (nilai/max( len(word1), len(word2)))  return hasil |

**Gambar 5.11 Tampilan *Source Code* Algoritma *Levenshtein Distance***

**5.3 Pengujian Sistem**

Pengujian merupakan tahap yang utama dalam pembuatan suatu aplikasi. Hasil dari pengujian yang didapat akan dijadikan sebagai tolak ukur dalam proses pengembangan selanjutnya.

**5.3.1 Pengujian *Black Box***

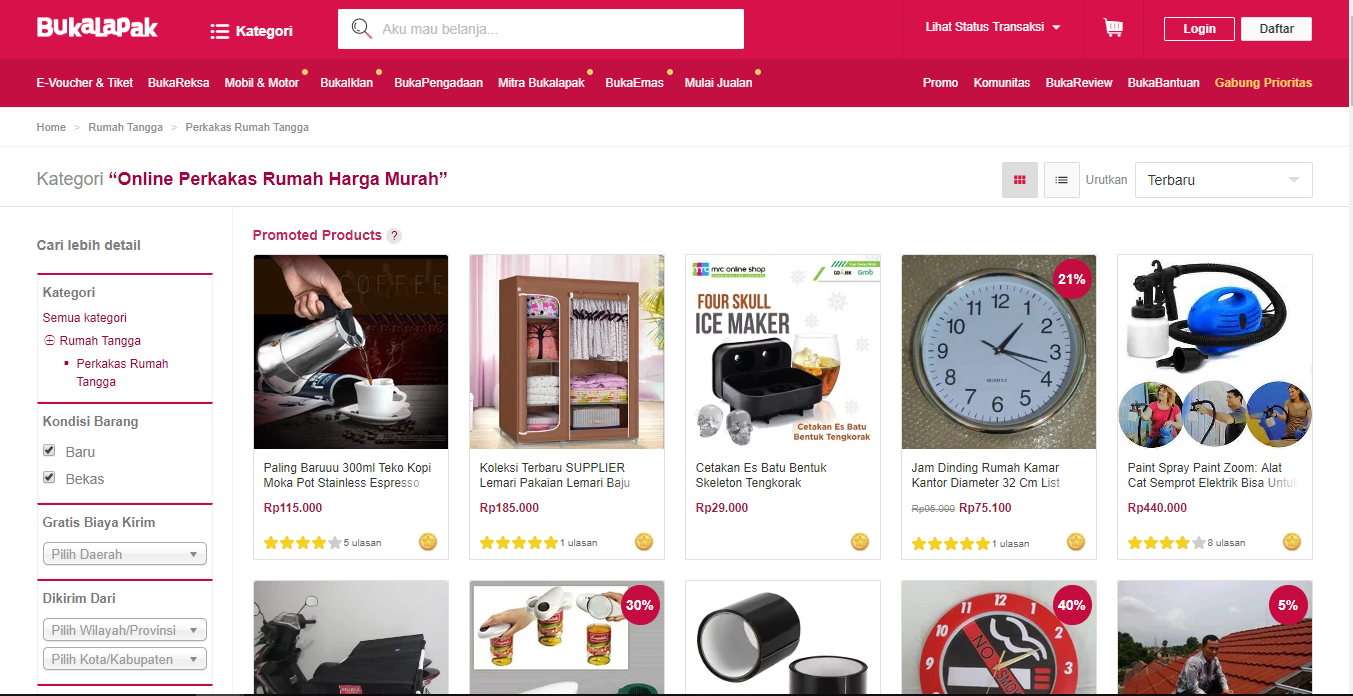
Pengujian *black box* dilakukan dengan menguji perangkat lunak dari segi fungsionalitas perangkat lunak. Fungsionalitas perangkat lunak yang diuji sesuai dengan *use case* pada tahap desain. Peneliti membagi pengujian menjadi lima bagian. Setiap bagian diuji sesuai dengan skenario *use case* pada tahap desain.

**Tabel 5.1 Pengujian *Black Box***

|  |  |  |  |
| --- | --- | --- | --- |
| Input/ Event | Proses | Output | Hasil Uji |
| Membuka *Website* | Proses masuk pada aplikasi pencarian barang | Menampilkan Menu Beranda | Sesuai |
| Memilih Menu Pencarian | Menampilkan data barang | Data barang ditampilkan | Sesuai |
| Memilih Menu Petunjuk | Menampilkan petunjuk penggunaan aplikasi | Petunjuk penggunaan aplikasi ditampilkan | Sesuai |
| Memilih Menu Tentang | Menampilkan Menu Tentang | Menampilkan tentang aplikasi | Sesuai |
| Pencarian Kata | Memasukkan kata yang ingin dicari dan menekan tombol *search* | Menampilkan hasil pencarian | Sesuai |

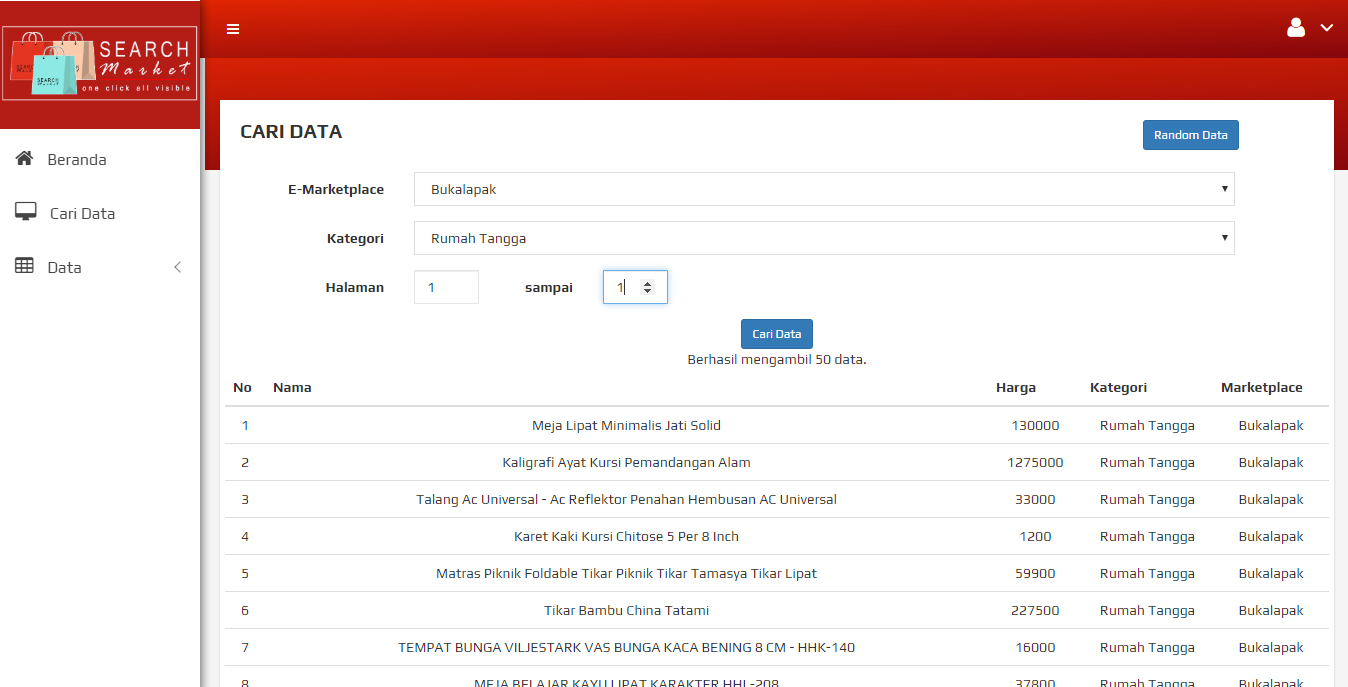
**5.3.2 Pengujian *Web Scraping* pada Bukalapak**

Pengujian ini dilakukan untuk memeriksa apakah sistem berhasil untuk melakukan pengambilan data pada *website* Bukalapak. Pengujian ini dilakukan pada *website* bukalapak pada kategori rumah tangga pada halaman satu terdapat 50 barang didalamnya seperti pada Gambar 5.12.



**Gambar 5.12 Pengujian pada Bukalapak**

Kemudian di sini akan melakukan proses pencarian data mengunakan fitur pencarian data menggunakan *web scraping* pada aplikasi seperti pada Gambar 5.13.

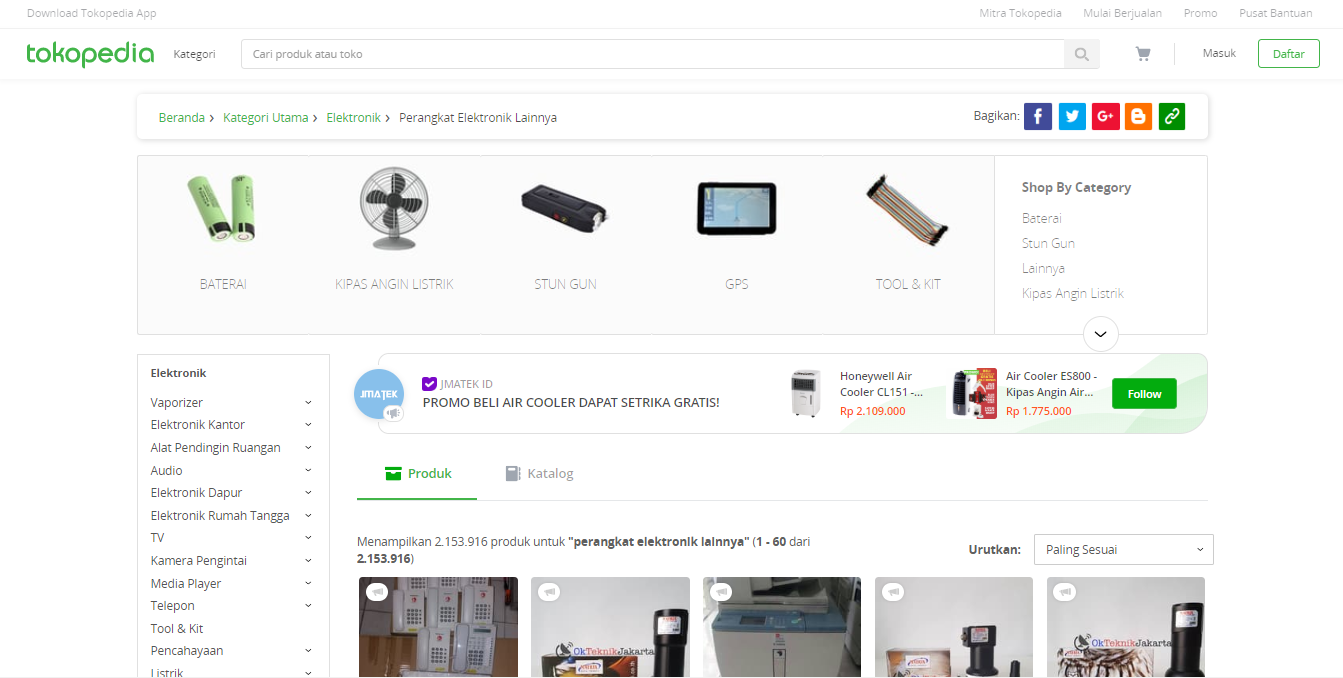


**Gambar 5.13 Pengujian *Web Scraping* pada Bukalapak**

Berdasarkan pengujian yang dilakukan pada *website* Bukalapak kategori rumah tangga halaman satu aplikasi berhasil mengambil seluruh data barang yaitu 50 barang yang terdapat pada *website* Bukalapak.

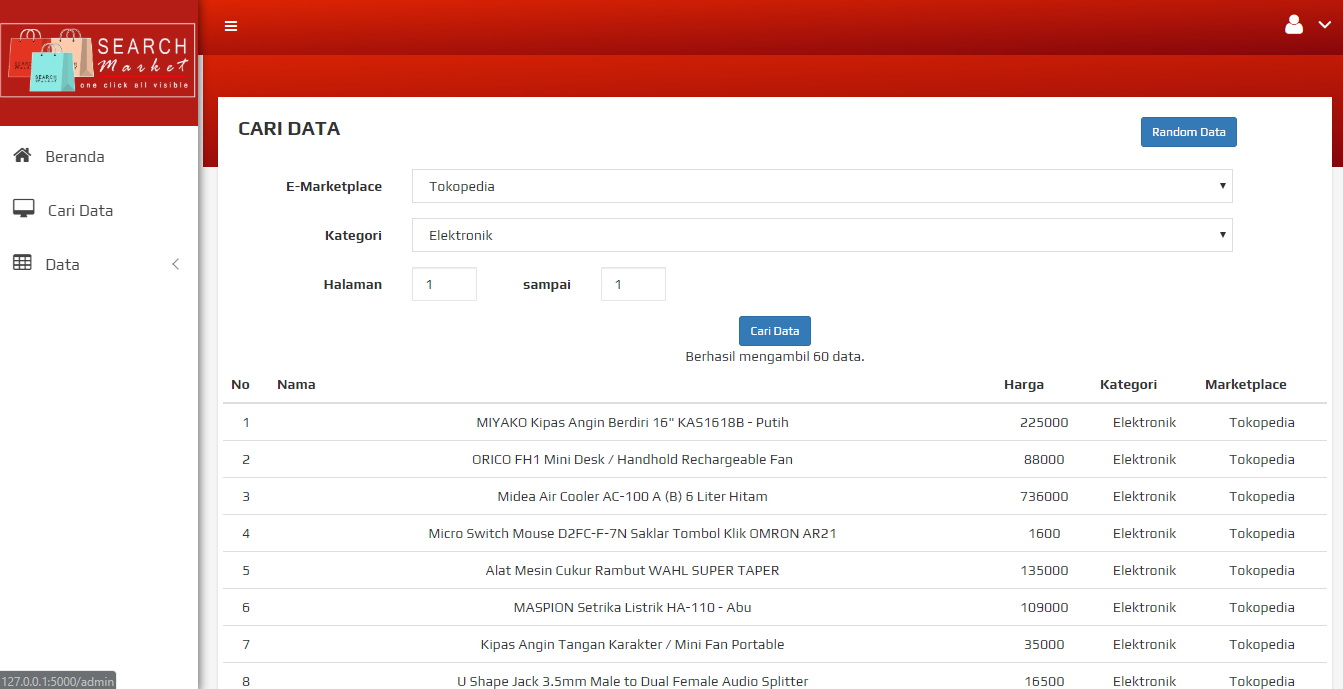
**5.3.3 Pengujian *Web Scraping* pada Tokopedia**

Pengujian ini dilakukan untuk memeriksa apakah sistem berhasil untuk melakukan pengambilan data pada *website* Tokopedia. Pengujian ini dilakukan pada *website* Tokopedia pada kategori elektronik pada halaman satu terdapat 60 barang didalamnya seperti pada Gambar 5.14.



**Gambar 5.14 Pengujian pada Tokopedia**

Kemudian di sini akan melakukan proses pencarian data mengunakan fitur pencarian data menggunakan *web scraping* pada aplikasi seperti pada Gambar 5.15.

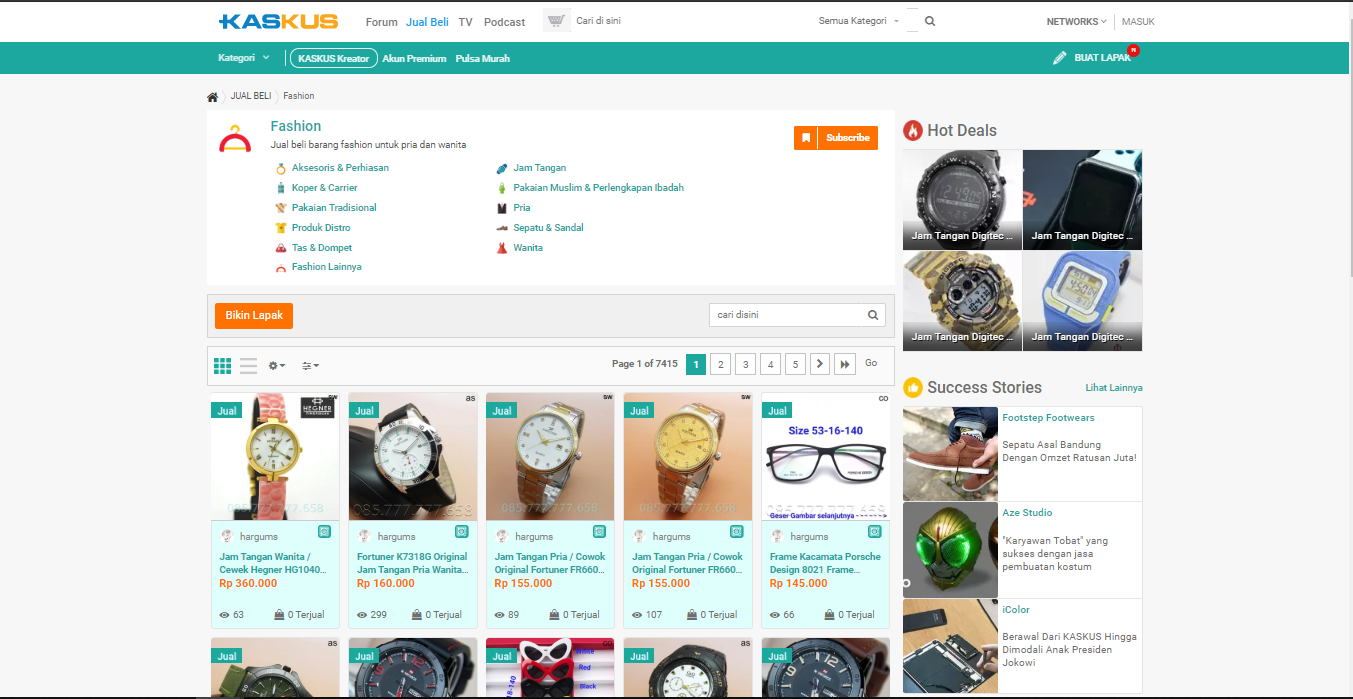


**Gambar 5.15 Pengujian *Web Scraping* pada Tokopedia**

Berdasarkan pengujian yang dilakukan pada *website* Tokopedia kategori elektronik halaman satu aplikasi berhasil mengambil seluruh data barang yaitu 60 barang yang terdapat pada *website* Tokopedia.

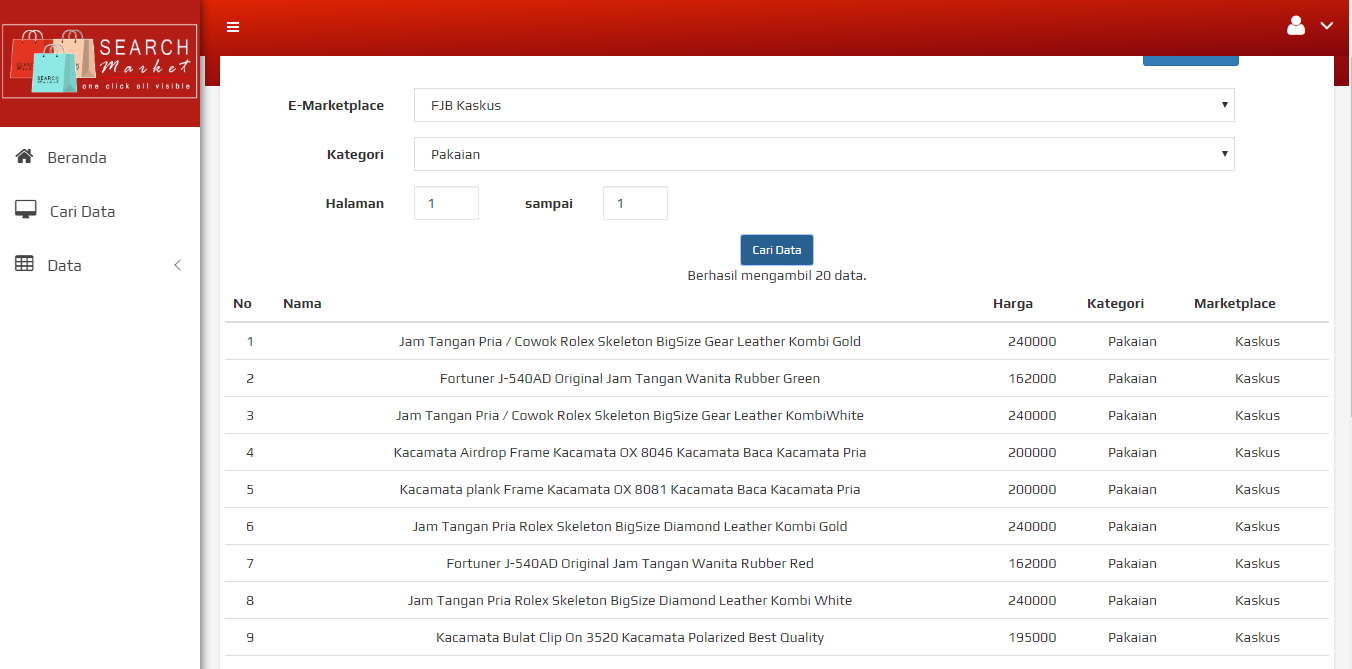
**5.3.4 Pengujian *Web Scraping* pada FJB Kaskus**

Pengujian ini dilakukan untuk memeriksa apakah sistem berhasil untuk melakukan pengambilan data pada *website* Tokopedia. Pengujian ini dilakukan pada *website* FJB Kaskus pada kategori *handphone* pada halaman satu terdapat 20 barang didalamnya seperti pada Gambar 5.16.



**Gambar 5.16 Pengujian pada FJB Kaskus**

Kemudian di sini akan melakukan proses pencarian data mengunakan fitur pencarian data menggunakan *web scraping* pada aplikasi seperti pada Gambar 5.17.



**Gambar 5.17 Pengujian *Web Scraping* pada FJB Kaskus**

Berdasarkan pengujian yang dilakukan pada *website* FJB Kaskus kategori handphone halaman satu aplikasi berhasil mengambil seluruh data barang yaitu 20 barang yang terdapat pada *website* FJB Kaskus.

**5.3.5 Pengujian Pencarian menggunakan *Levenshtein Distance***

Pengujian ini dilakukan untuk memeriksa apakah sistem berhasil melakukan pencarian menggunakan algoritma *levenshtein distance* dengan kata kunci yang dimasukkan oleh *user.* Pencarian ini dilakukan dengan cara *user* memasukkan kata kunci dan menekan tombol *search,* kemudian sistem akan melakukan mengecek semua kata yang terdapat pada *database*. Setelah itu sistem akan menghitung nilai kedekatan (*levenshtein distance*) antara kata kunci dan semua kata yang ada pada *database.* Kata yang memiliki nilai kedekatan lebih besar dari nilai batasan *levenshtein distance* akan ditampilkan sebagai hasil dari pencarian yang dilakukan oleh user.

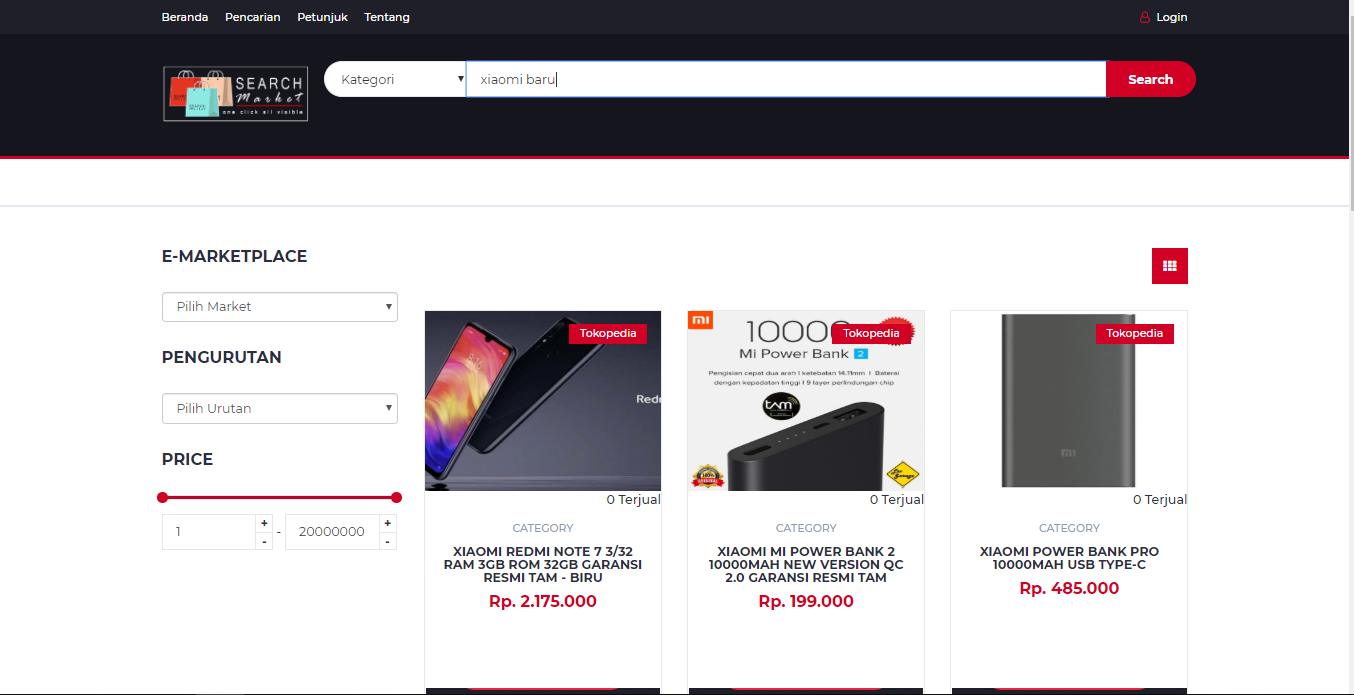
Untuk mendapatkan nilai batasan *levenshtein distance*, dilakukan pengujian nilai *levenshtein distance* pencarian pada *website* Search Market dengan memasukkan kata kunci untuk mencari barang yang diinginkan oleh pengguna. Pencarian ini dilakukan dengan satu kata atau lebih.

**Tabel 5.2 Pengujian Pencarian 1 Kata**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kata Pencarian** | **Judul Barang** | **Nilai** |
| 1 | xiaomi | Jual REDMI 4X Full Covered Color 2.5D Tempered Glass Xiaomi Redmi 4 X Temper Warna Full Cover Layar di lapak Yos Komsel yosafat85 | 1.0 |
| 2 | xiaomi | Jual Kabel OTG Micro USB Android BB Samsung Oppo Xiaomi On The Go di lapak King Kong Line kingkongline | 1.0 |
| 3 | xiaomi | Jual Kabel Data Oppo Original 2A Fast Charger F3 l F1s l A71 l A57 l A39 l A37 l Suport juga untuk Samsung Asus Xiaomi Vivo di lapak Alfa Store alfa\_store780 | 1.0 |
| 4 | xiaomi | Jual Sarung Tangan Harian Touring di lapak X M N xiamen | 0.5 |
| 5 | xiaomi | Jual serbuk kedelai hijau alami nasa di lapak Nasa Original megrosirnasa | 0.5 |
| 6 | xiaomi | Indomie MI GORENG 1 PCS | Indomi Mie Instan Noodles Goreng Murah Promo | 0.5 |

**Tabel 5.3 Pengujian Pencarian 2 Kata**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kata Pencarian** | **Judul Barang** | **Nilai** |
| 1 | xiaomi baru | Jual Kabel OTG Micro USB Android BB Samsung Oppo Xiaomi On The Go di lapak King Kong Line kingkongline | 0.64 |
| 2 | xiaomi baru | Mie Shirataki Biru - Wet Shirataki Blue Noodle 200 gram 10 bungkus | 0.54 |
| 3 | xiaomi baru | KHUSUS GOJEK Mie Shirataki Biru - Wet Shirataki Blue Noodle 200 gram | 0.54 |
| 4 | xiaomi baru | Mie Shirataki Biru untuk KETO DIET | 0.54 |



**Gambar 5.18 Pengujian Pencarian**

Dari tabel 5.5 dan 5.6 berhasil menampilkan barang yang akan digunakan pada percobaan selanjutnya dengan memilih kata yang mempengaruhi nilai *levenshtein distance* pada judul barang. Kata yang akan digunakan yaitu xiaomi, xiamen dan indomi. Kata-kata ini akan dibandingkan dengan kata xiaomi agar mendapatkan nilai rata-rata yang akan dijadikan sebagai nilai batasan *levenshtein distance*.

**Tabel 5.4 Pengujian Kata Pertama**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | - | x | i | a | o | m | i |
| - | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| x | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| i | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| a | 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| o | 4 | 3 | 2 | 1 | 0 | 1 | 2 |
| m | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| i | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Berdasarkan Tabel 5.4 telah didapatkan nilai levenshtein distance yaitu 0. Kemudian dengan nilai tersebut dilakukan perhitungan *similarity*.

Sim = 1 – () = 1

Sehingga mendapatkan nilai *similarity levenshtein distance* pada Tabel 5.4 adalah 1.

**Tabel 5.5 Pengujian Kata Kedua**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | - | x | i | a | o | m | i |
| - | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| x | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| i | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| a | 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| m | 4 | 3 | 2 | 1 | 1 | 1 | 2 |
| e | 5 | 4 | 3 | 2 | 2 | 2 | 2 |
| n | 6 | 5 | 4 | 3 | 3 | 3 | 3 |

Berdasarkan Tabel 5.5 telah didapatkan nilai levenshtein distance yaitu 3. Kemudian dengan nilai tersebut dilakukan perhitungan *similarity*.

Sim = 1 – () = 0.5

Sehingga mendapatkan nilai *similarity levenshtein distance* pada Tabel 5.5 adalah 0.5.

**Tabel 5.6 Pengujian Kata Ketiga**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | - | x | i | a | o | m | i |
| - | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| i | 1 | 1 | 1 | 2 | 3 | 4 | 5 |
| n | 2 | 2 | 2 | 2 | 3 | 4 | 5 |
| d | 3 | 3 | 3 | 3 | 3 | 4 | 5 |
| o | 4 | 4 | 4 | 4 | 3 | 4 | 5 |
| m | 5 | 5 | 5 | 5 | 4 | 3 | 4 |
| i | 6 | 6 | 5 | 6 | 5 | 4 | 3 |

Berdasarkan Tabel 5.6 telah didapatkan nilai levenshtein distance yaitu 3. Kemudian dengan nilai tersebut dilakukan perhitungan *similarity*.

Sim = 1 – () = 0.5

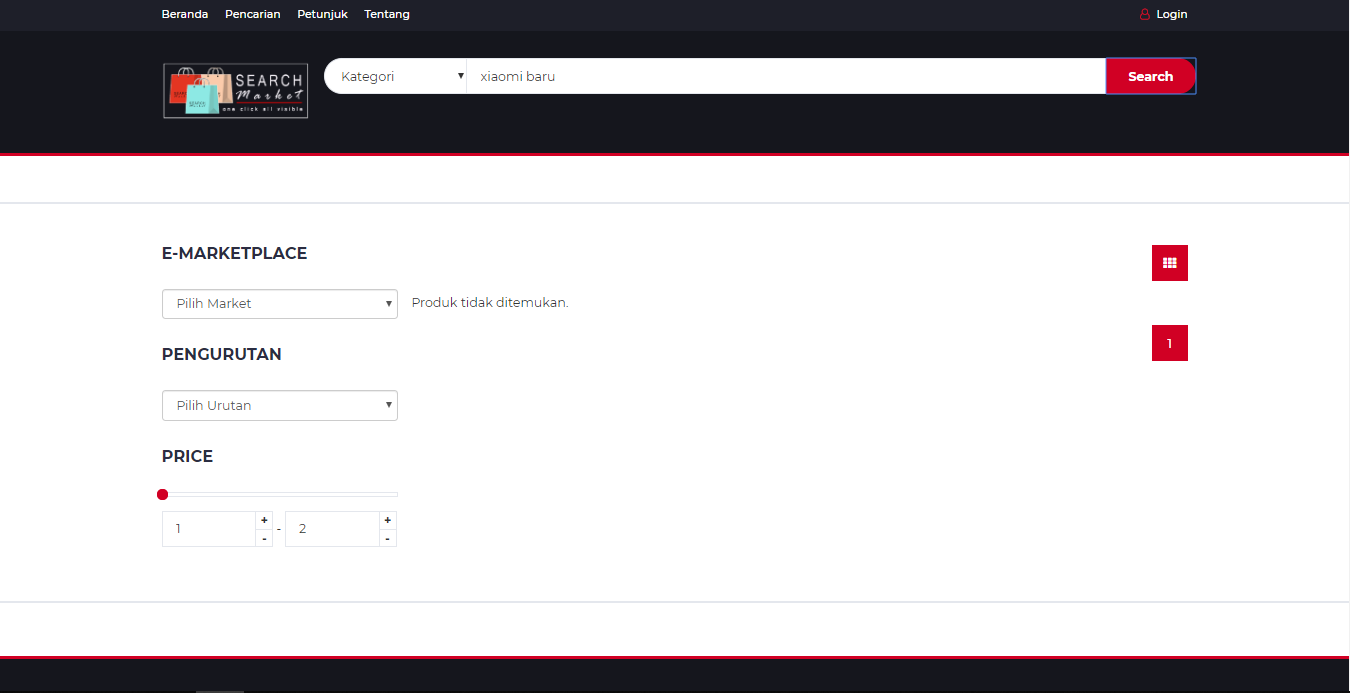
Sehingga mendapatkan nilai *similarity levenshtein distance* pada Tabel 5.6 adalah 0.5. Selanjutnya melakukan perhitungan rata-rata dari hasil percobaan tersebut.

= 0.6667

Jadi nilai batasan dari levenshtein distance adalah 0.6667 yang akan digunakan sebagai nilai minum dari pencarian pada sistem Search Market.

**5.3.6 Pengujian Pencarian Barang Tidak Ditemukan**

Pengujian ini dilakukan untuk memeriksa apakah yang ditampilkan oleh sistem apabila kata kunci yang dimasukkan oleh *user* tidak ada dalam *database.* Pengujian ini dilakukan dengan memasukkan kata kunci yang tidak terdapat pada *database* atau kata kunci yang memiliki kedekatan (*levenshtein distance*) lebih kecil dari 0.5 dari kata yang terdapat pada *database*.



**Gambar 5.19. Pengujian Pencarian Barang Tidak Ditemukan**

* 1. **Pengujian Kompleksitas Algoritma *Levenshtein Distance***

Pengujian kompleksitas algoritma *levenshtein distance* menggunakan *Notasi Big-O* yaitu untuk mengukur kompleksitas waktu yang diperlukan algoritma untuk menyelesaikan proses pencarian dalam kondisi terburuk. *Notasi Big-O* berfungsi dalam mengkategorikan algoritma ke fungsi yang menggambarkan *upper* *limit* atau batas atas dari pertumbuhan sebuah fungsi ketika masukan dari fungsi tersebut bertambah banyak dengan menggunakan fungsi *Big-O*. Disini nanti kompleksitas algoritma akan dinilai dengan *Notasi Big-O* yang terdiri dari *time complexity* atau berapa lama algoritma itu jalan, *space complexity* atau berapa banyak memori yang bakal dipakai oleh algoritma.

Pada algoritma yang memerlukan waktu yang konstan, menggandakan *problem size* tidak mempengaruhi jumlah operasi yang dilakukan. Apalagi ketika terdapat kasus terburuk, yang maksudnya terdapat jumlah operasi terbanyak yang mungkin dilakukan oleh algoritma untuk besaran masalah yang diberikan.

**Tabel 5.7 Kompleksitas Algoritma *Levenshtein Distance***

|  |  |  |  |
| --- | --- | --- | --- |
| Levenshtein distance | Lambang | | |
| C | # | C# |
| word1 = word1.lower() | C1 | 1 | C1 |
| word2 = word2.lower() | C1 | 1 | C1 |
| for x in range(len(word1) + 1): | C2 | N | C2N |
| matrix[x][0] = x | C2 | N | C2N |
| for y in range(len(word2)+1): | C3 | M | C3M |
| matrix[0][y] = y | C3 | M | C3M |
| for x in range(1, len(word1)+ 1): | C2 | N | C2N |
| for y in range(1, len(word2) + 1): | C4 | NM | C4NM |
| if word1[x - 1] == word2[y - 1]: | C4 | NM | C4NM |
| matrix[x][y] = min( | C4 | NM | C4NM |
| matrix[x - 1][y] + 1, | C4 | NM | C4NM |
| matrix[x - 1][y - 1], | C4 | NM | C4NM |
| matrix[x][y - 1] + 1 | C4 | NM | C4NM |
| }else | C4 | NM | C4NM |
| matrix[x][y] = min( | C4 | NM | C4NM |
| matrix[x - 1][y] + 1, | C4 | NM | C4NM |
| matrix[x - 1][y - 1] + 1, | C4 | NM | C4NM |
| matrix[x][y - 1] + 1  } | C4 | NM | C4NM |
| nilai = matrix[len(word1)][len(word2)] | C1 | 1 | C1 |
| hasil = 1 – (nilai/max( len(word1), len(word2))) | C1 | 1 | C1 |
| return hasil | C1 | 1 | C1 |

Keterangan :

C = Konstanta

# = Ukuran masukan

C . # = Kompleksitas waktu

N = Panjang kata 1

M = Panjang kata 2

TƟ = (3C1 + 3C2N + 2C3M + 11C4NM

= 0 + N + M + NM

= Ɵ(NM)

Tabel 5.2 adalah tabel kompleksitas Algoritma *Levenshtein Distance*, dimana proses pencarian kompleksitasnya menggunakan bahasa phyton, C sebagai konstanta, # sebagai ukuran masukan, dan C. # (C kali #) adalah untuk mencari *Theoritical Running Time* (T(n)) atau kompleksitas waktu, sehingga dapat dijumlahkan hasil dari perkalian C kali #, maka diperoleh hasil kompleksitas waktu yaitu θ(NM) yang artinya kompleksitas waktu Algoritma *Levenshtein Distance* sama dengan panjang kata 1 dikali panjang kata 2.

* 1. **Pengujian Penerimaan (*Acceptance Testing*)**

*Acceptance Testing* dilakukan dengan metode kuesioner, perhitungan hasil kuesioner dilakukan dengan menggunakan skala *likert.* Pengujian dilakukan dengan menjalankan sistem pencarian barang di berbagai *e-marketplace* dan diberikan form yang berisi sebelas pertanyaan kepada penguji aplikasi. Poin-poin pada pertanyaan tersebut berisi tentang fitur-fitur pada sistem aplikasi pencarian barang di berbagai *e-marketplace* apakah sesuai dengan permasalahan yang ada. Setiap pertanyaan memiliki 5 pilihan yaitu sangat setuju, setuju, netral, tidak setuju dan sangat tidak setuju. Pertanyaan dalam kuesoner tersebut sebagai berikut:

P1. Apakah Informasi yang disediakan oleh aplikasi ini mudah dimengerti?

P2. Apakah penggunaan menu atau fitur aplikasi menu mudah digunakan?

P3. Apakah aplikasi nyaman digunakan?

P4. Secara keseluruhan apakah penggunaan aplikasi ini memuaskan?

P5. Apakah aplikasi ini sesuai dengan kebutuhan?

P6. Apakah aplikasi dapat dengan mudah dipelajari?

P7. Apakah aplikasi mudah dioperasikan?

P8. Apakah dapat dengan mudah menghindari kesalahan dalam menggunakan aplikasi?

P9. Apakah aplikasi bermanfaat bagi pengguna?

P10. Apakah tampilan menu dalam aplikasi mudah untuk dikenali?

P11. Apakah aplikasi mempunyai kemampuan dan fungsi sesuai yang diharapkan?

Setelah melakukan survei dengan menyebarkan kuesioner berupa program aplikasi pencarian barang di berbagai *e-marketplace*, maka diperolehlah data sebanyak 75 responden. Tabel 5.3 berikut adalah hasil dari jawaban para responden mengenai kuesioner yang diberikan:

**Tabel 5.8 Hasil Jawaban Kuesioner**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pertanyaan | Jawaban Responden | | | | |
| Sangat Setuju (SS) | Setuju (S) | Netral (N) | Tidak Setuju (TS) | Sangat Tidak Setuju (STS) |
| P1 | 31 | 37 | 5 | 1 | 1 |
| P2 | 32 | 36 | 6 | 1 | 0 |
| P3 | 27 | 31 | 13 | 4 | 0 |
| P4 | 27 | 31 | 11 | 5 | 1 |
| P5 | 32 | 28 | 11 | 2 | 2 |
| P6 | 33 | 36 | 3 | 3 | 0 |
| P7 | 26 | 39 | 8 | 2 | 0 |
| P8 | 20 | 41 | 9 | 5 | 0 |
| P9 | 37 | 31 | 4 | 1 | 2 |
| P10 | 29 | 33 | 9 | 4 | 0 |
| P11 | 28 | 36 | 9 | 1 | 1 |
| Jumlah | 322 | 379 | 88 | 29 | 7 |

Dari data yang diperoleh tersebut kemudian dianalisis dengan menghitung rata-rata jawaban berdasarkan skor yang diperoleh dari setiap jawaban responden. Berdasarkan skor yang telah ditetapkan dapat dihitung sebagai berikut:

* Jumlah skor dari responden yang menjawab SS = 322 x 5 = 1610
* Jumlah skor dari responden yang menjawab S = 379 x 4 = 1516
* Jumlah skor dari responden yang menjawab N = 88 x 3 = 264
* Jumlah skor dari responden yang menjawab TS = 29 x 2 = 58
* Jumlah skor dari responden yang menjawab STS = 7 x 1 = 7
* Jumlah Total = 3397

Hasil jawaban dari responden sebanyak 75 orang tersebut di atas kemudian dapat dihitung nilai tertinggi dan terendah seperti berikut:

* Nilai tertinggi = 75 x 11 x 5 = 4125 (seandainya semua menjawab SS).
* Nilai terendah = 75 x 11 x 1 = 825 (seandainya semua menjawab STS).

Berdasarkan perhitungan yang menyatakan nilai tertinggi adalah 4125 dapat dicari persentase seperti berikut:

(3397 / 4125) x 100% = 82.35%.

Dari persentase tersebut kemudian dapat diketahui bahwa tingkat pengujian penerimaan (*Acceptance Testing*) sistem aplikasi pencarian barang di berbagai *e-marketplace* berdasarkan persepsi pengguna tergolong sangat setuju dengan hasil 82.35%. Hal ini dapat dilihat dari kriteria interpretasi skornya berdasarkan interval sebagai berikut:

* Angka 0% – 19,99% = Sangat (tidak setuju/buruk/kurang sekali)
* Angka 20% – 39,99% = Tidak Setuju / Kurang Baik
* Angka 40% – 59,99% = Cukup / Netral
* Angka 60% – 79,99% = Setuju/Baik/Suka
* Angka 80% – 100% = Sangat (setuju/baik/suka)