

**NO: 01022098/INF/2021**

**SISTEM PENUNJANG BELANJA PEDAGANG KELILING DI LOKASI SEKITAR MENGGUNAKAN  
*HAVERSINE BERBASIS ANDROID***

**SKRIPSI**

Diajukan untuk memenuhi persyaratan penyelesaian program S-1  
Program Studi Informatika Fakultas Teknologi Informasi  
Universitas Kristen Petra

Oleh:  
Richard Gozali  
NRP: C14170049

**PROGRAM STUDI INFORMATIKA**



**FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS KRISTEN PETRA  
SURABAYA  
2021**

## **LEMBAR PENGESAHAN**

### **SKRIPSI**

#### **SISTEM PENUNJANG BELANJA PEDAGANG KELILING DI LOKASI SEKITAR MENGGUNAKAN HAVERSINE BERBASIS ANDROID**

Oleh:

Richard Gozali C14170049

Diterima Oleh:

Program Studi Informatika  
Fakultas Teknologi Industri  
Universitas Kristen Petra

Surabaya, 28 Juni 2021

Dosen Pembimbing 1

Dosen Pembimbing 2

(Liliana,Ph.D).  
NIP: 03024

(Yulia,M.Kom)  
NIP: 99036

Ketua Tim Penguji:

(ANDREAS HANDOJO,M.MT).  
NIP: 00016  
Ketua Program Studi

(Henry Novianus Palit, S.Kom., M.Kom., Ph.D)  
NIP: 14001

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK  
KEPENTINGAN AKADEMIK**

Sebagai mahasiswa Universitas Kristen Petra, yang bertanda tangan dibawah ini saya:

Nama : Richard Gozali

NRP : C14170049

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Kristen Petra Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul "Sistem Penunjang Belanja Pedagang Keliling di Lokasi Sekitar Menggunakan *Haversine Berbasis Android*". Dengan Hak Bebas Royalti Non Eksklusi ini, Universitas Kristen Petra berhak menyimpan, mengalihmediakan/formatkan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikan nya, menampilkan/mempublikasikannya di internet atau media lain untuk kepentingan akademik tanpa perlu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

Kami bersedia untuk menanggung secara pribadi, tanpa melibatkan Universitas Kristen Petra, segala bentuk tuntutan hukum yang timbul atau Pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian pernyataan saya ini yang saya buat dengan sebenarnya.

Surabaya, 28 Juni 2021

Yang menyatakan,

Richard Gozali

## KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya yang telah penulis terima selama melaksanakan tugas akhir ini, sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini.

Tugas akhir ini diajukan untuk memenuhi salah satu persyaratan guna mencapai gelar Sarjana Strata-1 di Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra, Surabaya.

Penulisan skripsi ini juga tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada berbagai pihak yang telah membantu penulis selama ini, khususnya:

1. Liliana,Ph.D. selaku dosen pembimbing 1 yang telah memberikan bimbingan dan pengarahan dalam penyusunan Tugas Akhir ini.
2. Yulia,M.Kom. selaku dosen pembimbing 2 yang telah memberikan bimbingan dan pengarahan dalam penyusunan Tugas Akhir ini.
3. ANDREAS HANDOJO, S.T., M.MT.. selaku ketua tim penguji.
4. ANDRE GUNAWAN, S.Kom. selaku dosen penguji 2.
5. Henry Novianus Palit, S.Kom., M.Kom., Ph.D selaku Ketua Program Studi Informatika Universitas Kristen Petra Surabaya.
6. Segenap dosen dan *staff* pengajar di Program Studi Informatika Universitas Kristen Petra Surabaya.
7. Para pedagang keliling dan para pelanggan yang telah membantu dalam pembuatan skripsi dengan menjadi objek penelitian dan melakukan survey yang telah diberikan.
8. Keluarga yang telah banyak memberikan dukungan doa dan motivasi.
9. Pihak-pihak lain yang telah memberikan bantuan secara langsung maupun tidak langsung dalam pembuatan skripsi ini yang tidak dapat disebutkan satu per satu.

Penulisan menyadari bahwa penulisan skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan segala petunjuk, kritik, dan saran yang membangun dari pembaca agar dapat menunjang pengembangan dan perbaikan penulisan selanjutnya.

Akhir kata penulis mohon maaf apabila ada kekurangan dalam penulisan tugas akhir ini dan penulis dengan senang hati menerima saran dan kritik yang membangun dari pembaca. Semoga penyajian Tugas Akhir ini bisa memberikan manfaat kepada berbagai pihak lain.

Surabaya, 28 Juni 2021

Richard Gozali

## ABSTRACT

Richard Gozali :

Skripsi

Sistem Penunjang Belanja Pedagang Keliling di Lokasi Sekitar Menggunakan Haversine Berbasis Android.

Pedagang keliling merupakan usaha yang dilakukan oleh seseorang untuk menjual barang/jasa seperti sayur, buah, dan lain-lain dengan cara berkeliling ke tempat-tempat dengan rute tertentu. Banyaknya kebutuhan para pelanggan untuk mendapatkan barang yang diinginkan dan kemudahan untuk melakukan pembelanjaan. Salah satu bisnis yang tumbuh di masa pandemi Covid-19 adalah retail, terutama yang melayani kebutuhan sehari-hari seperti sayur-sayuran dan buah-buahan. Dikarenakan pada masa pandemi ini sebagian orang membeli produk melalui online. Lonjakan permintaan ini membuat banyak orang mencari penjual sayur-sayuran dan buah-buahan.

Dengan adanya aplikasi ini diharapkan mampu membantu pedagang keliling untuk mendapatkan *market* dan juga sistem yang mudah untuk menjalankan aplikasi. Dengan adanya aplikasi ini diharapkan juga pelanggan dapat mengetahui lokasi dan keberadaan pedagang keliling di sekitar. Penggunaan rumus *haversine* untuk mengetahui jarak antara pelanggan dan pedagang keliling memudahkan untuk pelanggan mengetahui jarak pedagang keliling. Penggunaan *scraping* yang dilakukan pada situs *eresep.com* diperlukan untuk mendapatkan menu makanan, cara memasak dan cara pembuatannya. Dengan menggunakan *BeautifulSoup* data menu makanan dapat di ekstraksi.

Untuk pembuatan aplikasi dibutuhkanya server untuk menyimpan data lokasi pedagang keliling, pelanggan, produk pedagang keliling, menu makanan dan orderan. Setelah itu dilakukan proses untuk mengolah data tersebut dan membuat UI dari aplikasi tersebut. Berdasarkan hasil pengujian, program yang dibuat telah berhasil untuk mengumpulkan menu masakan, lokasi pedagang keliling dan pelanggan, jarak pedagang keliling dan pelanggan, membuat sistem pembelian antara pedagang keliling dan pelanggan dan Membuat fitur komunikasi antara pedagang keliling dan pelanggan. Meskipun masih ada kekurangan fitur dalam aplikasi tersebut.

Kata Kunci :

*Web Scraping, BeautifulSoup, Kotlin, Android, Haversine*

## **ABSTRACT**

Richard Gozali :

Undergraduate Thesis

traveling merchants Shopping Support System in Nearby Locations Using Android Based Haversine.

A traveling merchant is an effort made by a person to sell goods/services such as vegetables, fruit, and others by traveling to places with certain routes. The number of customer needs to get the desired goods and convenience to make purchases. One of the businesses that grew during the Covid-19 pandemic was retail, especially those serving daily needs such as vegetables and fruits. Due to this pandemic, some people buy products online. This surge in demand has made many people look for sellers of vegetables and fruits.

With this application, it is expected to be able to help traveling merchants to get to the market and also an easy system to run the application. With this application, it is also hoped that customers can find out the location and whereabouts of traveling merchants around. The use of the haversine formula to determine the distance between customers and traveling merchants makes it easier for customers to know the distance of traveling merchants. The use of scraping carried out on the erep.com site is intended to get a food menu, how to cook and how to make it. By using BeautifulSoup food menu data can be extracted.

For making applications, a server is needed to store data on the location of traveling merchants, customers, mobile merchant products, food menus and orders. After that, a process is carried out to process the data and create the UI of the application. Based on the test results, the program created has been successful in gathering food menus, the location of traveling merchants and customers, the distance between traveling merchants and customers, creating a purchasing system between traveling merchants and customers and Creating communication features between traveling merchants and customers. Although there is still a lack of features in the application.

Keywords :

*Web Scraping, BeautifulSoup, Kotlin, Android, Haversine*

## DAFTAR ISI

SISTEM PENUNJANG BELANJA PEDAGANG KELILING DI LOKASI SEKITAR MENGGUNAKAN <i>HAVERSINE BERBASIS ANDROID</i> .....	i
LEMBAR PENGESAHAN.....	ii
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIK .....	iii
KATA PENGANTAR.....	iv
ABSTRACT.....	vi
ABSTRACT.....	vii
DAFTAR ISI .....	viii
DAFTAR GAMBAR.....	xi
DAFTAR TABLE.....	xiii
DAFTAR SEGMENT PROGRAM .....	xiv
BAB 1. PENDAHULUAN .....	1
1.1    Latar Belakang Masalah .....	1
1.2    Perumusan Masalah.....	2
1.3    Tujuan Skripsi .....	2
1.4    Ruang Lingkup .....	2
1.5    Metodologi Penelitian.....	4
1.6    Sistematika Penulisan.....	5
BAB 2. LANDASAN TEORI.....	6
2.1    Haversine Formula .....	6
2.2    Web Scraping .....	6
2.3    Text Mining.....	7
2.4    Grabmart dan Gomart.....	7
2.5    Tinjauan Studi.....	8
2.5.1    Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route .....	8
BAB 3. ANALISIS DAN DESAIN SISTEM.....	9
3.1    Analisis Data .....	9
3.2    Desain Arsitektur.....	9
3.3    Analisis Sistem.....	10

3.4	Analisis Permasalahan.....	10
3.5	Use Case Diagram.....	10
3.5.1	Entity Relation Diagram .....	11
3.5.2	Penjelasan Struktur Tabel .....	13
3.6	Proses <i>Scraping</i> .....	17
3.6.1	Proses Tracking Pedagang Keliling .....	19
3.6.2	Proses List Pedang Keliling dan Search Produk.....	20
3.6.3	Proses Menambah Produk Pedagang Keliling.....	22
3.6.4	Proses Pembelian Barang.....	23
3.7	<i>Desain Interface</i> Pelanggan.....	23
3.7.1	Tampilan <i>Login</i> Pelanggan.....	24
3.7.2	Tampilan <i>Registrasi</i> Pelanggan.....	25
3.7.3	Tampilan <i>Home</i> Pelanggan.....	26
3.7.4	Tampilan <i>Menu</i> Makanan Pelanggan.....	27
3.7.5	Tampilan Aktivitas Pelanggan .....	28
3.7.6	Tampilan Detail Pedagang.....	29
3.8	Desain Interface Pedagang Keliling .....	30
3.8.1	Desain Home Pedagang Keliling.....	30
3.8.2	Desain Aktivitas Pedagang Keliling.....	31
BAB 4. IMPLEMENTASI SISTEM .....		32
4.1	Proses pembuatan <i>API</i> melalui <i>php</i> dan <i>mysql</i> .....	32
4.2	Proses <i>Scraping</i> Menu Makanan.....	32
4.3	Proses List Pedagang Keliling .....	36
4.4	Proses Pembelian Produk.....	41
4.5	Proses Upload Produk .....	50
Bab 5 PENGUJIAN SISTEM .....		55
5.1	Pengujian Data Produk Pedagang Keliling.....	55
5.2	Pengujian Fitur Pelanggan.....	56
5.2.1	Pengujian <i>Tracking</i> Pedagang Keliling .....	56
5.2.2	Pengujian <i>Scraping</i> Menu Makanan.....	60
5.2.3	Pengujian Filter Pedagang Keliling Region .....	65
5.2.4	Pengujian Fitur Pembelian Produk.....	66
5.3	Pengujian Fitur Pedagang Keliling .....	68

5.3.1	Pengujian Upload Produk Pedagang Keliling .....	68
5.3.2	Pengujian Fitur Otomatis Tambah Barang .....	69
5.3.3	Pengujian Fitur Chatting.....	71
5.4	Hasil KuesionerPelanggan .....	73
5.5	Hasil Kuesioner Pedagang Keliling.....	75
	BAB 6 KESIMPULAN DAN SARAN .....	78
6.1	Kesimpulan .....	78
6.2	Saran.....	79

## DAFTAR GAMBAR

Gambar 3.1 Arsitektur Sistem dari Aplikasi .....	9
Gambar 3.2 <i>Use Case Diagram</i> , .....	11
Gambar 3.3 <i>Entity Relation Diagram</i> .....	12
Gambar 3.4 <i>Entity Relation Diagram Physical</i> .....	13
Gambar 3.5 <i>Activity Diagram</i> Proses <i>Scraping</i> Menu Makanan.....	18
Gambar 3.6 Flowchart <i>Scraping</i> Makanan.....	19
Gambar 3.7 <i>Activity Diagram</i> <i>Tracking</i> Pedagang Keliling.....	20
Gambar 3.8 <i>List</i> Pedagang Keliling .....	21
Gambar 3.9 <i>Search</i> Produk.....	21
Gambar 3.10 Menambah Barang .....	22
Gambar 3.11 Proses Pembelian Produk.....	23
Gambar 3.12 Tampilan Login Pelanggan.....	24
Gambar 3.13 Tampilan Registrasi Pelanggan.....	25
Gambar 3.14 Tampilan Home Pelanggan.....	26
Gambar 3.15 Tampilan Menu Makanan .....	27
Gambar 3.16 Tampilan Menu Aktivitas Pelanggan .....	28
Gambar 3.17 Tampilan Detail Pelanggan.....	29
Gambar 3.18 Tampilan Detail Pelanggan .....	29
Gambar 3.19 Tampilan Home Pedagang Keliling .....	30
Gambar 3.20 Tampilan Aktivitas Pedagang Keliling.....	31
Gambar 5.1 Database Produk Pedagang.....	55
Gambar 5.2 Pengujian <i>Tracking</i> Pedagang Keliling .....	56
Gambar 5.3 Pengujian <i>Tracking</i> Pedagang Keliling Kedua .....	57
Gambar 5.4 Pengujian <i>Tracking</i> Pedagang Keliling Ketiga .....	57
Gambar 5.5 Perbandingan Jarak <i>Haversine</i> dan <i>Google Maps 1</i> .....	58
Gambar 5.6 Jarak Menggunakan <i>Google Maps 1</i> .....	59
Gambar 5.7 Perbandingan Jarak Haversine dan Google Maps 2 .....	59
Gambar 5.8 Jarak Menggunakan Google Maps 2 .....	60
Gambar 5.9 Pengujian <i>Scraping</i> 1 .....	61
Gambar 5.10 Pengujian <i>Scraping</i> 2 .....	62
Gambar 5.11 Hasil Pengujian <i>Scraping</i> 1 .....	63
Gambar 5.12 Hasil Pengujian <i>Scraping</i> 2 .....	63

Gambar 5.13 <i>Database</i> Menu Masakan Sebelum <i>Scraping</i> .....	64
Gambar 5.14 <i>Database</i> Menu Masakan Setelah <i>Scraping</i> .....	64
Gambar 5.15 Hasil Filter Lokasi Region Pedagang Keliling.....	65
Gambar 5.16 Hasil <i>Filter</i> Lokasi <i>Region</i> Pedagang Keliling.....	66
Gambar 5.17 Pembelian Produk Pedagang Keliling .....	67
Gambar 5.18 Data Pembelian Produk di Pedagang Keliling.....	67
Gambar 5.19 Tampilan Upload Produk.....	68
Gambar 5.20 Pengujian Tambah Barang Otomatis 2.....	69
Gambar 5.21 Pengujian Tambah Barang Otomatis.....	70
Gambar 5.22 Pengujian Tambah Barang Otomatis.....	70
Gambar 5.23 Tampilan <i>Detail Order</i> .....	71
Gambar 5.24 Hasil <i>chatting</i> dan <i>notifikasi</i> .....	72
Gambar 5.25 Hasil <i>Chatting</i> dan <i>Notifikasi</i> Pelanggan .....	72

## DAFTAR TABLE

Tabel 2.1 Fitur - Fitur Grab dan Gomart.....	7
Tabel 3.1 Desain Tabel Pelanggan.....	14
Tabel 3.2 Desain Tabel Menu Masakan .....	14
Tabel 3.3 Desain Tabel Menu Makanan Pelanggan .....	15
Tabel 3.4 Desain Pedagang Keliling.....	15
Tabel 3.5 Desain Tabel Produk .....	16
Tabel 3.6 Desain Tabel Produk Pedagang .....	16
Tabel 3.7 Desain Tabel Detail Produk.....	16
Tabel 3.8 Desain Tabel Order .....	17
Tabel 4.1 Daftar Hubungan <i>activity diagram</i> dan segmen program.....	32
Tabel 5.1 Pengujian Waktu <i>Scraping</i> Menu Masakan.....	65
Tabel 5.2 Waktu Upload Produk .....	69
Tabel 5.3 Hasil Kuesioner Pelanggan.....	73
Tabel 5.4 Hasil Kuesioner Pedagang Keliling.....	75

## **DAFTAR SEGMENT PROGRAM**

Segmen Program 4.1 database.php .....	32
Segmen Program 4.2 Proses scraping situs <i>eresep.com</i> .....	33
Segmen Program 4.3 Proses List Pedagang Keliling.....	37
Segmen Program 4.4 Pembelian Produk.....	42
Segmen Program 4.5 Upload Produk.....	50

## BAB 1. PENDAHULUAN

Pada bab ini akan dijelaskan mengenai latar belakang masalah, perumusan masalah, tujuan, ruang lingkup, metodologi penelitian, dan juga sistematika penulisan dari skripsi yang dibuat.

### 1.1 Latar Belakang Masalah

Pedagang keliling merupakan usaha yang dilakukan oleh seseorang untuk menjual barang/jasa seperti sayur, buah, dan lain-lain dengan cara berkeliling ke tempat-tempat dengan rute tertentu. Kebutuhan sayur mayur, buah serta bahan baku lauk pauk lainnya dibutuhkan untuk kebutuhan sehari -hari. Banyak masyarakat lebih memilih tukang sayur keliling daripada pergi ke pasar atau berbelanja di tempat keramaian (2020, April 26). Pada masa karantina secara tidak langsung membuat orang - orang menjadi lebih kreatif dalam mengolah makan (2021, January 21.). Pandemi *Covid-19* serta kebiasaan tinggal di rumah saja, membuat –banyak orang lebih memilih memasak sendiri dibandingkan dengan membeli (2020, July 21).

Salah satu bisnis yang tumbuh di masa pandemi Covid-19 adalah retail, terutama yang melayani kebutuhan sehari-hari seperti sayur-sayuran dan buah-buahan. Dikarenakan pada masa pandemi ini sebagian orang membeli produk melalui online. Lonjakan permintaan ini membuat banyak orang mencari penjual sayur-sayuran dan buah-buahan (2020, April 25). Adanya aplikasi untuk tukang sayur keliling ini dapat membantu banyak orang dalam menemukan tukang sayur keliling pada area sekitar. Aplikasi besar seperti Gojek dan Grab tidak memasukan pedagang sayur keliling kedalam sistem aplikasinya. Dengan adanya aplikasi ini diharapkan mampu untuk membantu para pedagang sayur keliling menjadi lebih mudah dan efisien.

Pada penelitian sebelumnya yaitu "*Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route*" memiliki perkiraan tempat terdekat untuk pergi ketempat yang dituju. Di penelitian pada skripsi ini, saya ditambahkan fitur perkiraan waktu. Melalui fitur ini, para pelanggan dapat mengetahui waktu sampai dari pedagang keliling.

Masalah yang diangkat dari penelitian ini adalah kesusahanya para pedagang keliling untuk mendapatkan market dan sistem yang dapat membantu pedagang keliling. Dengan adanya aplikasi ini diharapkan dapat membantu pedagang keliling untuk lebih memperluas dagangan dan memudahkan para pengguna aplikasi ini untuk membeli produk dengan lebih mudah dan cepat. Dengan menggunakan rumus haversine diharapkan dapat membantu para pelanggan untuk

mendapatkan lokasi terdekat dari pedagang keliling dan membuat pengantar produk yang dibeli lebih cepat dan lebih efisien dibandingkan dengan cara yang biasa saja.

## **1.2 Perumusan Masalah**

Perumusan masalah dari skripsi ini adalah :

1. Bagaimana cara membuat sistem yang memudahkan pedagang keliling untuk memenuhi kebutuhan belanja dari masyarakat yang berada di sekitar lokasi ?
2. Apakah rumus haversine dapat membantu aplikasi ini untuk mendapatkan ketepatan jarak yang sesuai ?
3. Apakah aplikasi dapat dipakai dengan mudah oleh pedagang keliling dan pelanggan ?

## **1.3 Tujuan Skripsi**

Tujuan dari sistem ini diharapkan mampu menghasilkan sistem yang dapat membantu para pelanggan pedagang keliling untuk berbelanja dengan cepat dan efisien. Mampu memudahkan para pedagang keliling untuk mendapatkan market dan juga memberikan sistem yang mudah untuk para pedagang keliling.

## **1.4 Ruang Lingkup**

Ruang Lingkup dibatasi pada :

1. *Knowledge Base dataset* untuk letak posisi pedagang keliling. Data yang digunakan menggunakan longitude dan latitude.
2. Aplikasi yang digunakan untuk *backend* adalah bahasa pemrograman Python. *API* menggunakan bahasa pemrograman PHP. *Android* menggunakan bahasa pemrograman Kotlin. Algoritma yang digunakan adalah *Haversine Formula*.
3. *Database* yang digunakan merupakan *MySQL* dari *server* tos.petra.ac.id
4. Melakukan survei terhadap 5 pedagang keliling.
5. Fitur – fitur yang dibuat :
  - Memiliki dua tampilan berbeda yaitu pedagang keliling dan pelanggan. Dimana di pedagang keliling mendapat 3 halaman. Yang pertama: halaman barang - barang yang telah terdaftar, Kedua halaman: aktivitas pada pedagang keliling dengan pelanggan, Ketiga halaman user settings. Pada halaman pertama pelanggan terdapat search pedagang keliling. Halaman kedua resep setelah itu terdapat tampilan list resep. Halaman

ketiga : terdapat aktivitas dan tampilan area pedagang keliling di sekitar area. Halaman keempat : terdapat *user setting*.

- Pada saat melakukan pendaftaran pedagang keliling ini akan melakukan pendaftaran tempat secara daerah.
- Fitur Pelanggan : Pengorderan produk untuk sehari sebelumnya untuk pedagang keliling dan pengorderan untuk hari – H dengan menggunakan posisi region dari pedagang keliling.
- Fitur Pelanggan : Pembayaran melalui bayar di tempat atau COD.
- Fitur Pelanggan : Dapat menentukan siapa pedagang kelilingnya yang berada di dalam daerah dengan menggunakan rumus haversine.
- Pedagang keliling dan pelanggan bisa melakukan *chatting*
- Dibuat sistem pemesanan online via Android
- Bisa melakukan pencarian posisi dari pedagang sayur keliling
- Fitur Pelanggan : Melakukan proses scraping resep masakan dari Eresep.com dan menu makanan tersimpan di dalam database

6. *Output* yang diberikan posisi dari pedagang sayur keliling, rekomendasi buah-buahan dan sayuran bagi user.

7. Target:

a. Demografis :

i. Jenis Kelamin: Pria dan Wanita

ii. Usia : 13 tahun - 50 Tahun

b. Geografis : Kota Surabaya

c. *Behavioral* :

1. Pengguna aplikasi *Android*

2. Ingin membeli produk dagang keliling dengan mudah

d. Psikologis : Mau untuk mencoba hal yang baru

8. Pengujian tingkat keberhasilan : Melakukan survei terhadap masyarakat dan pedagang keliling tentang fitur-fitur dan kemudahan fitur. Untuk menguji lokasi pedagang keliling ke masyarakat dengan metode haversine

## **1.5 Metodologi Penelitian**

Langkah – langkah dalam pengerajan skripsi :

1. Studi literatur tentang:
  - 1.1. Survey dengan pedagang keliling
  - 1.2. Teori Mobile Development berbasis Android
2. Pembuatan aplikasi :
  - 2.1. Menggunakan bahasa kotlin.
  - 2.2. Membuat API untuk koneksi database
3. Pengujian dan Analisis
  - 3.1. Pengujian sistem pedagang keliling berbasis Android
  - 3.2. Analisis hasil dari program
- 4 Pengambilan Kesimpulan
  - 4.1 Membandingkan hasil yang dihasilkan dengan kondisi jawaban yang seharusnya.

## **1.6 Sistematika Penulisan**

### Bab 1: Pendahuluan

Bab ini berisi latar belakang masalah, rumusan masalah, tujuan penelitian, ruang lingkup, metodologi penelitian, dan tinjauan pustaka.

### Bab 2: Landasan Teori

Bab ini berisi teori penunjang yang digunakan dalam proses pembuatan skripsi.

### Bab 3: Desain dan Analisa Sistem

Bab ini berisi tentang perencanaan sistem, meliputi desain interface dari aplikasi yang akan dibuat, dan desain rule dari sistem aplikasi yang akan dibuat.

### Bab 4: Implementasi Sistem

Bab ini berisi tahap implementasi desain dan analisis sistem ke dalam bentuk aplikasi dan program.

### Bab 5: Pengujian Program

Bab ini berisi hasil dari pengujian aplikasi dan program scraping yang telah dibuat, dengan menggunakan tahap-tahap yang telah dijelaskan pada bab-bab sebelumnya.

### Bab 6: Kesimpulan dan Saran

Bab ini berisi pembahasan mengenai kesimpulan yang didapat dan saran yang bermanfaat untuk pengembangan dari sistem atau program dan aplikasi yang telah dibuat.

## BAB 2. LANDASAN TEORI

Pada bab ini akan dijelaskan teori-teori yang akan digunakan dalam Sistem Penunjang Belanja Pedagang Keliling di Lokasi Sekitar Menggunakan Haversine Berbasis Android. Adapun teori-teori yang akan dijelaskan adalah teori tentang *Haversine Formula* , *Web Scraping*, *Text Mining* , Grabmart dan Gomart.

### 2.1 Haversine Formula

*Haversine Formula* merupakan sebuah algoritma yang dibuat untuk mengatur jarak terdekat menggunakan letak lokasi *latitude* dan *longitude* (2020, April 25).

$$r = 2.R.\arcsin \left\{ \sqrt{\sin^2 \left( \frac{\text{lat}_x - \text{lat}_y}{2} \right)} + \cos(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \sin^2 \left( \frac{\text{long}_x - \text{long}_y}{2} \right)} \right\}$$

1. lat x = Latitude dari koordinat pertama
2. lat y = Latitude dari koordinat kedua
3. long x = Longitude dari koordinat pertama
4. long y = Longitude dari koordinat kedua
5. R = Radius Bumi 6.371 Km
6. r = jarak yang dihasilkan

### 2.2 Web Scraping

Website adalah sekumpulan halaman pada suatu domain di internet yang dibuat dengan tujuan tertentu dan saling berhubungan dapat diakses secara langsung melalui halaman depan (home page) menggunakan sebuah browser dengan menggunakan *URL*. Website pertama kali dibuat oleh Tim Berners-Lee pada akhir 1980 dan baru resmi online pada tahun 1991. Tujuan awal website adalah memudahkan peneliti untuk bertukar informasi atau melakukan perubahan informasi (2020, November 27).

*Web scraping* adalah teknik untuk melakukan ekstraksi data dan informasi dari suatu website kemudian menyimpannya dalam format tertentu. Biasanya scraping ini bisa dilakukan salah satunya untuk mendapatkan produk atau barang yang berada di situs tertentu. Teknik web scraping bisa dilakukan dengan manual atau otomatis menggunakan tools. Parsing HTML adalah teknik yang menggunakan JavaScript untuk menargetkan halam linear *HTML* dan nested *HTML*. Teknik parsing ini bisa dengan lebih cepat melakukan identifikasi *script HTML* (2020, March 31).

*BeautifulSoup* merupakan *library* dari *Python* yang diperuntukan untuk web scraping yang menggunakan *package bs4*. Digunakan untuk ekstraksi tipe file *XML* atau *HTML*.

### 2.3 Text Mining

*Text Mining* adalah mengambil data berupa teks atau kata - kata dimana sumber data didapatkan dari dokumen, Bertujuan untuk mendapatkan kata - kata yang diinginkan. Proses ekstraksi terhadap pola berupa informasi yang berguna dari sejumlah besar sumber data teks. *Text Mining* dianggap proses dua tahap diawali dengan penerapan struktur terhadap sumber data dan dilanjutkan dengan ekstraksi informasi dan pengetahuan yang relevan dari data teks terstruktur ini dengan menggunakan teknik dan alat yang sama dengan penambangan data.

Tujuan dari *text mining* adalah mendapatkan informasi yang berguna dari sekumpulan dokumen. Sumber data yang digunakan adalah kumpulan teks yang memiliki format yang tidak terstruktur. Tugas khusus yang dimiliki antara lain yaitu *text categorization* dan *text clustering*. Text mining juga menerapkan konsep dan teknik *data mining* untuk mencari pola teks (2016, December 15).

### 2.4 Grabmart dan Gomart

Merupakan aplikasi *ondemand* yang menyediakan produk seperti belanjaan, makanan kemasan, produk perawatan kesehatan, produk kecantikan dan hadiah. Gomart merupakan layanan pesan antar dimana pelanggan memesan produk belanjaan sesuai aplikasi lalu, *Mitra Driver* akan mengambil pesanan yang dipesan di *outlet* yang dituju.

Tabel 2.1 Fitur - Fitur Grab dan Gomart

Fitur	Gomart	Grabmart	Pedagang keliling
Melakukan pemesanan secara online pada hari-hari	X	X	X
Dapat melakukan search produk sesuai dengan produk yang dibutuhkan	X	X	X
Pemesanan produk sebelum hari-hari			X
Pembayaran Produk dengan sistem COD	X	X	X
Dapat menentukan pedagang keliling dengan mengetahui jarak			X
Melakukan pembelian produk di pasar atau swalayan	X	X	
Memiliki sorting produk bedasarkan kategori		X	X
Memiliki Promo tiap harinya	X	X	
Pembayaran Produk dengan payment gateway dan emoney	X	X	
Memiliki fitur love untuk menyimpan toko dan barang yang dinginkan		X	
Memiliki fitur lokasi area terdekat	X	X	X
Menunjukkan lokasi dan posisi dan produk			X
Menunjukkan Toko sebanyak 10 tempat terdahulu lalu di perbanyak setelah di scroll kebawah	X	X	X
Melakukan rating terhadap toko,barang dan driver	X	X	

## **2.5 Tinjauan Studi**

### **2.5.1 Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route**

Pada *paper* “*Implementation Of Haversine Formula And Best First Search Method In Searching Of Tsunami Evacuation Route*”. Menggunakan rumus *haversine* sebagai penentuan jarak yang tsunami dari lokasi yang didapat dan diberikan daerah yang aman untuk tsunami. Di dalam *paper* ini juga memberikan fitur untuk mengetahui jarak yang dapat oleh pengguna kemudian akan diberikan perkiraan waktu tsunami yang akan datang(Swara, G. Y. 2017). Pada skripsi yang dibuat juga menggunakan perhitungan *haversine* untuk menentukan jarak – jarak dari pedagang keliling dan juga menentukan batasan jarak dari pedagang keliling

### BAB 3. ANALISIS DAN DESAIN SISTEM

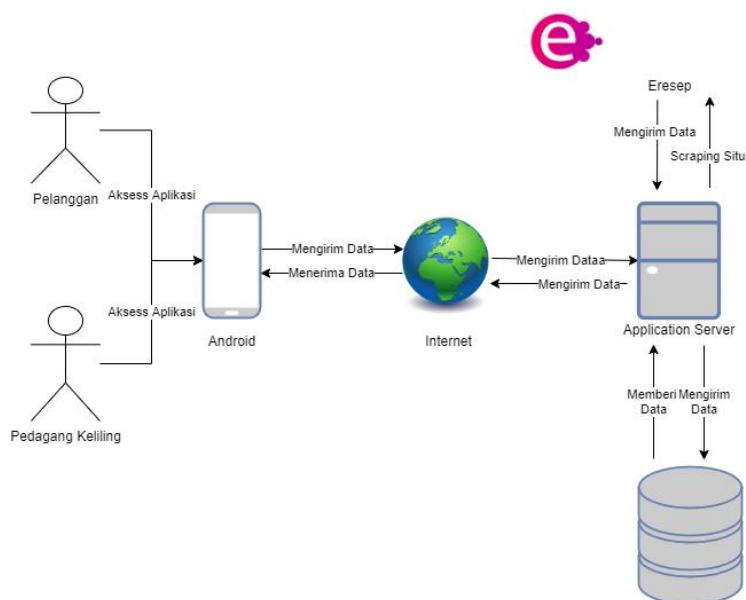
Pada bab ini akan dibahas mengenai desain dan analisis sistem mengenai Sistem Penunjang Belanja Pedagang Keliling di Lokasi Sekitar Menggunakan *Haversine* Berbasis *Android*. Pada bab ini pula akan dijelaskan analisis data yang akan membahas mengenai *usecase diagram*, *activity diagram* dan *user interface* yang dipakai.

#### 3.1 Analisis Data

Selama pembuatan skripsi, dilakukan analisis data terhadap gomart dan grabmart sebagai studi banding. Data yang digunakan dalam pembuatan sistem ini pertama adalah lokasi atau koordinat *latitude* dan *longitude* dari pengguna pedagang keliling dan pengguna. Data kedua yang diberikan adalah file gambar berupa *file jpg/jpeg* untuk produk dari pedagang keliling. Yang digunakan untuk mengupload gambar produk ke *server*. Data yang selanjutnya yang dibutuhkan adalah data pribadi dari pedagang keliling yang berupa *region/daerah* yang dibutuhkan untuk menghasilkan query bagi pelanggan. Data selanjutnya *real time location* pelanggan yang digunakan untuk mendapat list dari pedagang keliling terdekat di areanya.

#### 3.2 Desain Arsitektur

Untuk Mengetahui bagaimana suatu sistem bekerja, dibutuhkan sebuah arsitektur sistem. Yang bertujuan agar pengguna bisa memahami proses yang terjadi. Merupakan Gambar dari Gambar 3.1 Arsitektur Sistem dari Aplikasi keseluruhan aplikasi



Gambar 3.1 Arsitektur Sistem dari Aplikasi

### **3.3 Analisis Sistem**

Pada penulisan skripsi ini dilakukan analisis terhadap aplikasi sistem aplikasi gojek dan gomart untuk melihat – lihat fitur yang telah ada. Dan mencari tau fitur yang dapat ditambah dan digunakan dalam pembuatan sistem penunjang belanja pedagang keliling di lokasi sekitar.

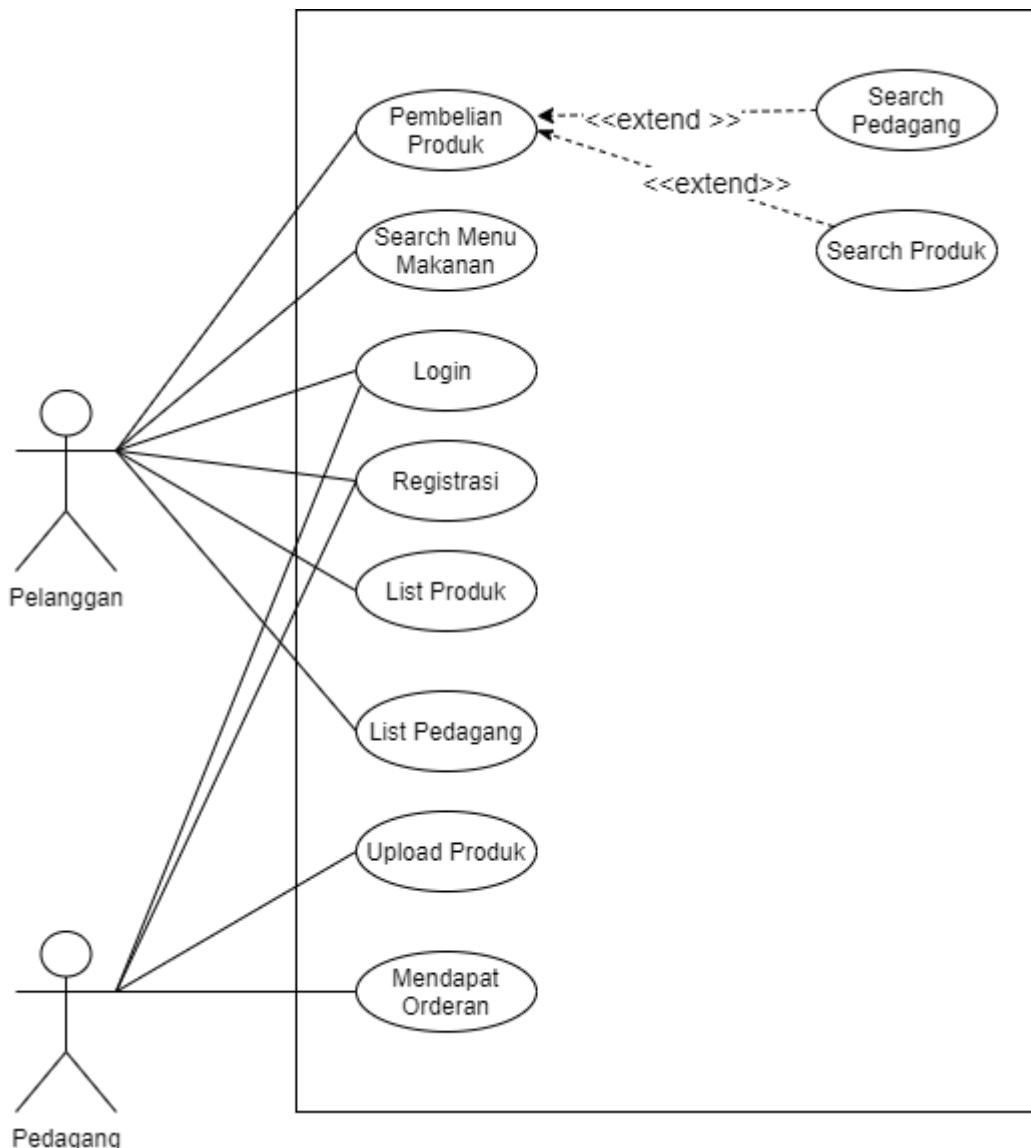
### **3.4 Analisis Permasalahan**

Kesusahannya pedagang keliling untuk menjual barangnya secara manual dan kesusahan pada pedagang keliling untuk mengetahui siapa saja yang ingin membeli barang daganganya sehingga pedagang keliling tidak usah untuk putar –putar untuk mendapatkan pelanggannya. Diharapkan juga membantu para pelanggan pedagang keliling untuk mengetahui lokasi pedagang keliling atau penjual yang dibutuhkan.

Membantu pelanggan dalam pembelian hari – H dan memberikan catatan untuk pembelian di esok paginya. Kesusahannya para pelanggan untuk menemukan pedagang keliling yang berada disekitar dan mengetahui produk-produk apa saja yang dibawa oleh pedagang keliling. Dengan adanya fitur pembelian membantu pedagang keliling untuk dapat menghafal pembelian apa saja yang di dapat oleh pelanggan. Pelanggan dapat melakukan pemesanan secara mudah dan dapat melakukan pemesanan dengan memiliki hari yang dinginkan. Para pelanggan kesusahan juga dalam mengetahui posisi *realtime* pedagang keliling.

### **3.5 Use Case Diagram**

*Use case diagram* merupakan *diagram* yang menggambarkan sebuah aktor dengan sistem. Use case diagram dapat mendeskripsikan satu *actor* atau lebih dari sebuah sistem. Di dalam use case diagram dalam penelitian ini terdapat dua aktor. Pelanggan yang memiliki hak kegunaan *login*, *registrasi*, *search produk*, *search pedagang keliling*, melihat *list* pedagang keliling dan melakukan pesanan produk. Pedagang keliling memiliki hak akses *login*, *registrasi*, *upload produk*, mendapat pesanan.



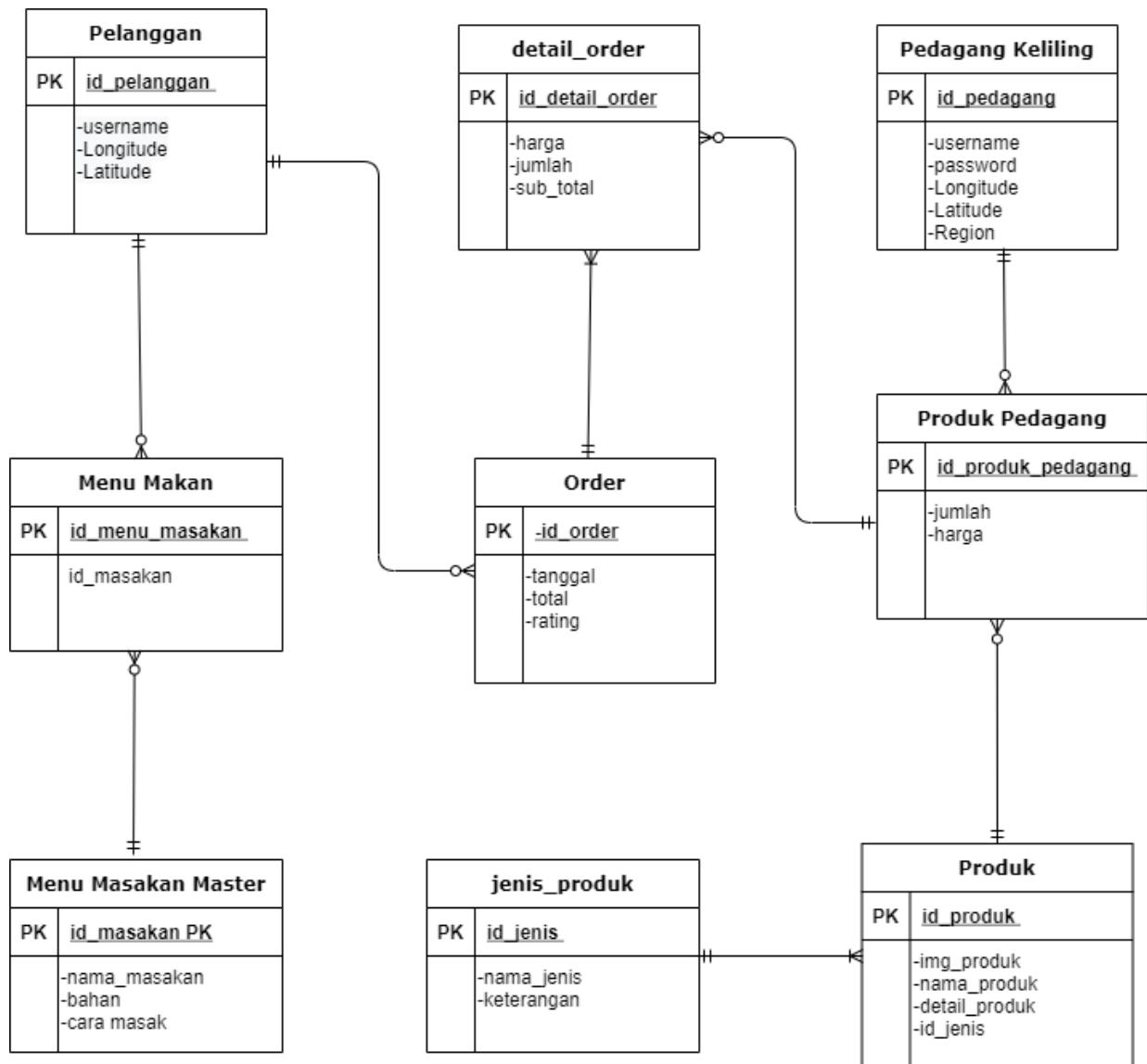
Gambar 3.2 Use Case Diagram,

### 3.5.1 Entity Relation Diagram

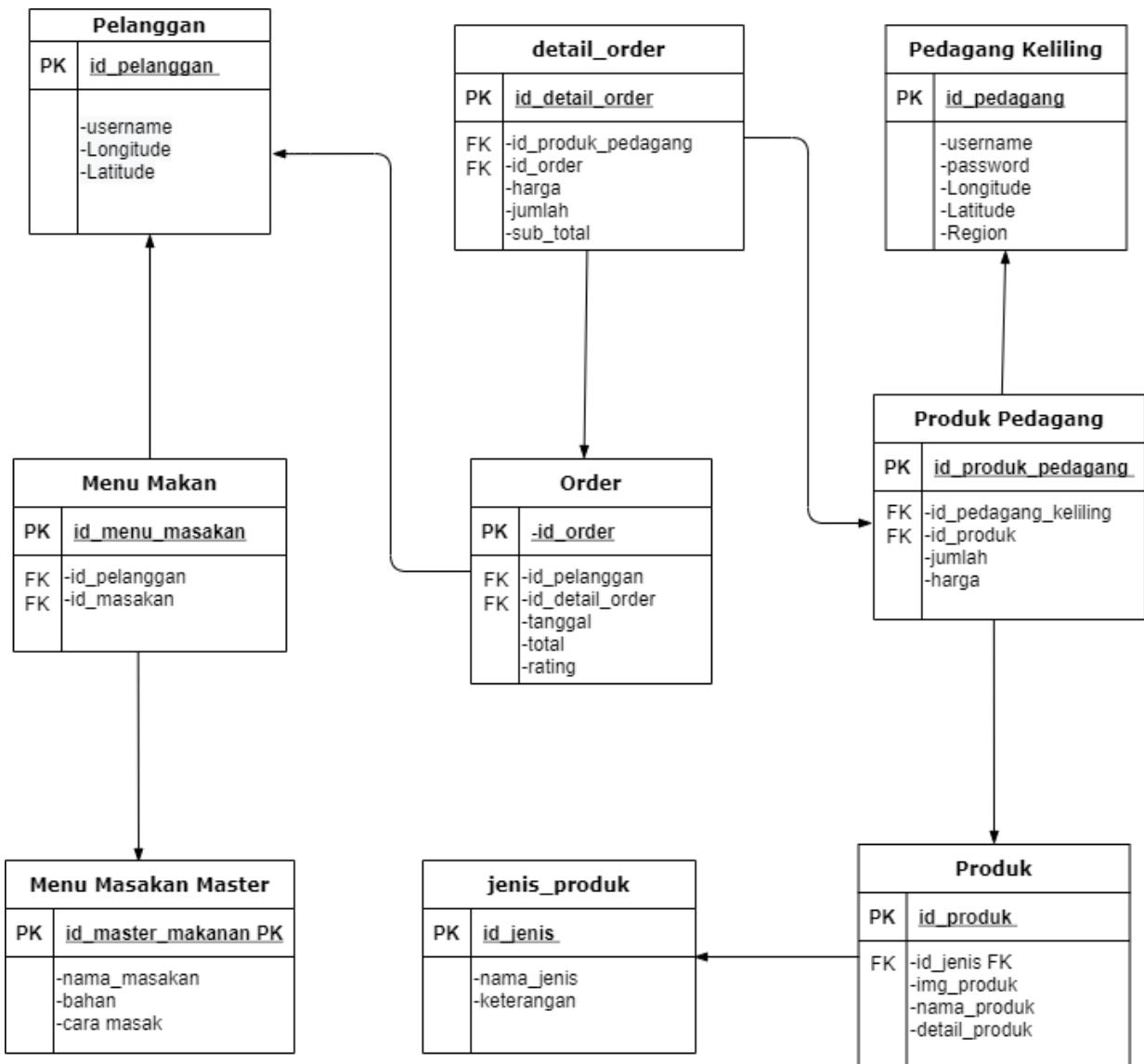
Pada penelitian ini memiliki 5 entity. Entity Pelanggan yang memiliki *attribute* `id_pelanggan`, `username`, `password`, `longitude` dan `latitude`. Entity Pedagang Keliling memiliki *attribute* `id_pedanggang`, `username`, `password`, `longitude`, `latitude`, `region`. Entity Produk `id_produk`, `foreign key id_pedagang`, `img_produk`, `nama_produk`, `jumlah_produk` dan `detail_produk`.

*Entity* keranjang memiliki *attribute* `id_keranjang`, `foreign key id_produk` dan `foreign key id_pelanggan`. *Entity* Order memiliki *attribute* `id_order` dan `id_kerajang`. Pelanggan memiliki

relation 1 to many ke pedagang keliling. Pedagang keliling memiliki relasi 1 to *many* ke produk. Pelanggan memiliki relasi 1 to *many* ke keranjang. Produk memiliki relasi 1 to *many* ke keranjang. Keranjang memiliki relasi 1 to 1 ke order.



Gambar 3.3 Entity Relation Diagram



Gambar 3.4 Entity Relation Diagram Physical

### 3.5.2 Penjelasan Struktur Tabel

Berikut ini adalah penjelasan struktur tabel yang ada di dalam *database* aplikasi. Berdasarkan *ERD* (*Entity Relationship Diagram*) yang sudah didesain sesuai pada Gambar 3.3, dibuat *database* dengan rincian tabel yang digunakan seperti dibawah ini.

Tabel 3.1 Desain Tabel Pelanggan

Nama	Tipe Data	Key	Deskripsi
id_pelanggan	Int	Primary key	Id Pelanggan auto increment
nama_pelanggan	Varchar(100)	-	nama pelanggan
username	Varchar(100)	-	username pelanggan
password	Varchar(100)	-	password pelanggan
longitude	Decimal	-	Longitude pelanggan
latitude	Decimal	-	Latitude pelanggan
rating	float	-	rating pelanggan

Desain tabel pelanggan pada Tabel 3.1 Desain Tabel Pelanggan digunakan untuk menyimpan data pelanggan yang ada. Data yang disimpan berupa username, password, nama\_pelanggan, longitude dan latitude dari pelanggan.

Tabel 3.2 Desain Tabel Menu Masakan

Nama	Tipe Data	Key	Deskripsi
id_menu_masakan	Int	Primary key	Id menu masakan auto increment
nama_masakan	Varchar(100)	-	nama masakan
bahan_masakan	Varchar(100)	-	bahan makanan
cara_memasak	Varchar(100)	-	cara- cara membuat makanan
pembuat_masakan	Varchar(100)	-	pembuat menu masakan

Desain Tabel Menu Masakan pada Tabel 3.2 Desain Tabel Menu Masakan digunakan untuk menyimpan data masakan yang di *scraping*. Data yang disimpan berupa nama\_masakan, bahan\_masakan, cara\_memasak, dan pembuat masakan.

Tabel 3.3 Desain Tabel Menu Makanan Pelanggan

Nama	Tipe Data	Key	Deskripsi
id_menu_makanana_user	Int	<i>Primary key</i>	Id menu masakan auto increment
id_pelanggan	int	<i>Foreign key</i>	id_pelanggan
id_menu_masakan	int	<i>Foreign key</i>	id_menu masakan dari tabel menu masakan

Desain Tabel Menu Masakan pada Tabel 3.3 Desain Tabel Menu Makanan Pelanggan digunakan untuk menyimpan data menu\_makanan user. Data yang disimpan berupa id\_pelanggan dan id\_menu\_masakan. Yang keduanya merupakan foreign key.

Tabel 3.4 Desain Pedagang Keliling

Nama	Tipe Data	Key	Deskripsi
id_pedagang_keliling	Int	<i>Primary key</i>	Id pedagang keliling auto increment
nama_pedagang_keliling	Varchar(100)	-	nama pedagang keliling
username	Varchar(100)	-	username pedagang keliling
password	Varchar(100)	-	password pedagang keliling
longitude	Decimal		Longitude pedagang keliling
latitude	Decimal		Latitude pedagang keliling
region_latitude	Decimal		Region latitude pedagang keliling
region_longitude	Decimal		Region longitude pedagang keliling

Desain tabel pelanggan pada Tabel 3.4 Desain Pedagang Keliling digunakan untuk menyimpan data pedagang keliling yang ada. Data yang disimpan berupa username, password, nama\_pedagang\_keliling, longitude, latitude, region longitude dan region latitude.

Tabel 3.5 Desain Tabel Produk

Nama	Tipe Data	Key	Deskripsi
id_produk	Int	<i>primary key</i>	Id produk auto increment
img_produk	Varchar(200)		Menyimpan link berupa gambar.
nama_produk	Varchar(100)		Nama_produk
detail_produk	Varchar(255)		Menyimpan keterangan produk
id_jenis	int	<i>foreign key</i>	foreign key dari tabel jenis_produk

Desain tabel produk pada Tabel 3.5 Desain Tabel Produk digunakan untuk menyimpan produk – produk yang di data. Data yang disimpan berupa img\_produk , nama\_produk , detail\_produk, id\_jenis.

Tabel 3.6 Desain Tabel Produk Pedagang

Nama	Tipe Data	Key	Deskripsi
id_produk_pedagang	Int	<i>primary key</i>	Id produk pedagang auto increment
id_produk	int	<i>foreign key</i>	merupakan foreign key dari produk
id_pedagang_keliling	int	<i>foreign key</i>	foreign key dari pedagang keliling
jumlah	int		jumlah produk yang tersedia
harga	float		harga dari produk

Desain tabel produk pedagang pada Tabel 3.6 Desain Tabel Produk Pedagang digunakan untuk menyimpan barang dagangan yang dimiliki oleh pedagang keliling. Data yang disimpan berupa id\_produk, id\_pedagang\_keliling, jumlah dan harga.

Tabel 3.7 Desain Tabel Detail Produk

Nama	Tipe Data	Key	Deskripsi
id_detail_order	Int	<i>primary key</i>	Id detail order auto increment
id_produk_pedagang	Int	<i>foreign key</i>	foreign key dari tabel produk_pedagang
id_order	int	<i>foreign key</i>	id_order
jumlah_detail_order	int		jumlah barang yang diorder
sub_total	float		total harga dari barang yang diorder

Tabel 3.7 Desain Tabel Detail Produk merupakan tabel yang menyimpan keranjang barang yang ingin dipesan. Digunakan untuk menentukan barang apa saja yang ingin dibeli. Data yang disimpan berupa id\_detail\_order, id\_produk\_pedagang, jumlah\_detail\_order, dan subtotal.

Tabel 3.8 Desain Tabel Order

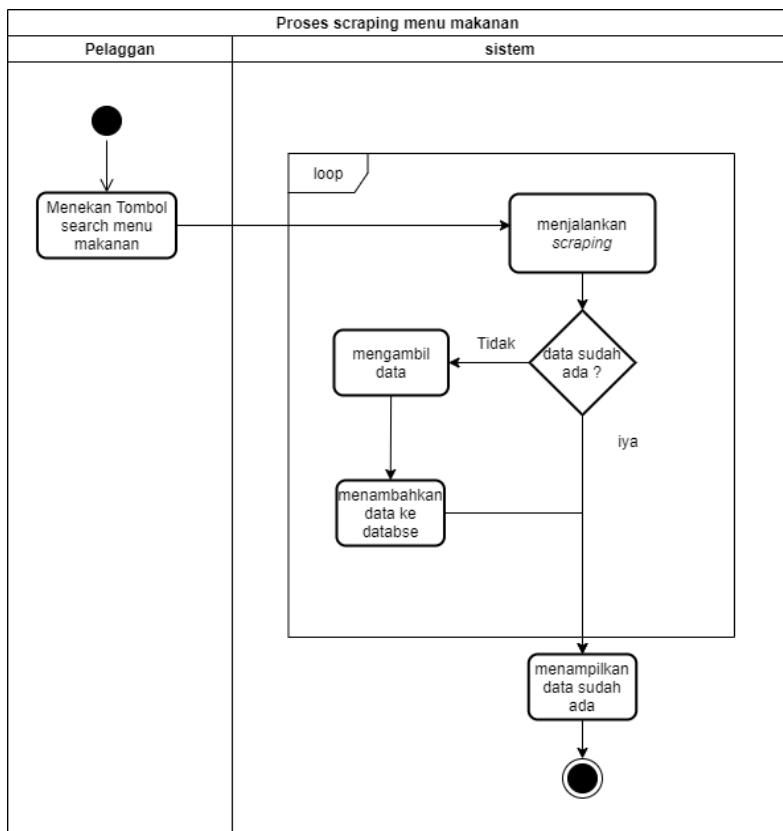
Nama	Tipe Data	Key	Deskripsi
id_order	Int	<i>primary key</i>	id order auto increment
id_detail_order	Int	<i>foreign key</i>	foreign key dari tabel produk_pedagang
tanggal	date		menyimpan tanggal berapa barang dipesan
total	float		total harga dari produk yang diorder
rating	float		menyimpan rating produk dan pelanggan
id_pelanggan	int	<i>foreign key</i>	foreign key dari tabel pelanggan

Tabel 3.8 Desain Tabel Order digunakan untuk menyimpan data pembelian pelanggan dan juga untuk menampilkan history pembelian. Data ini menyimpan id\_detail\_order, tanggal , total dan id \_pelanggan

### 3.6 Proses *Scraping*

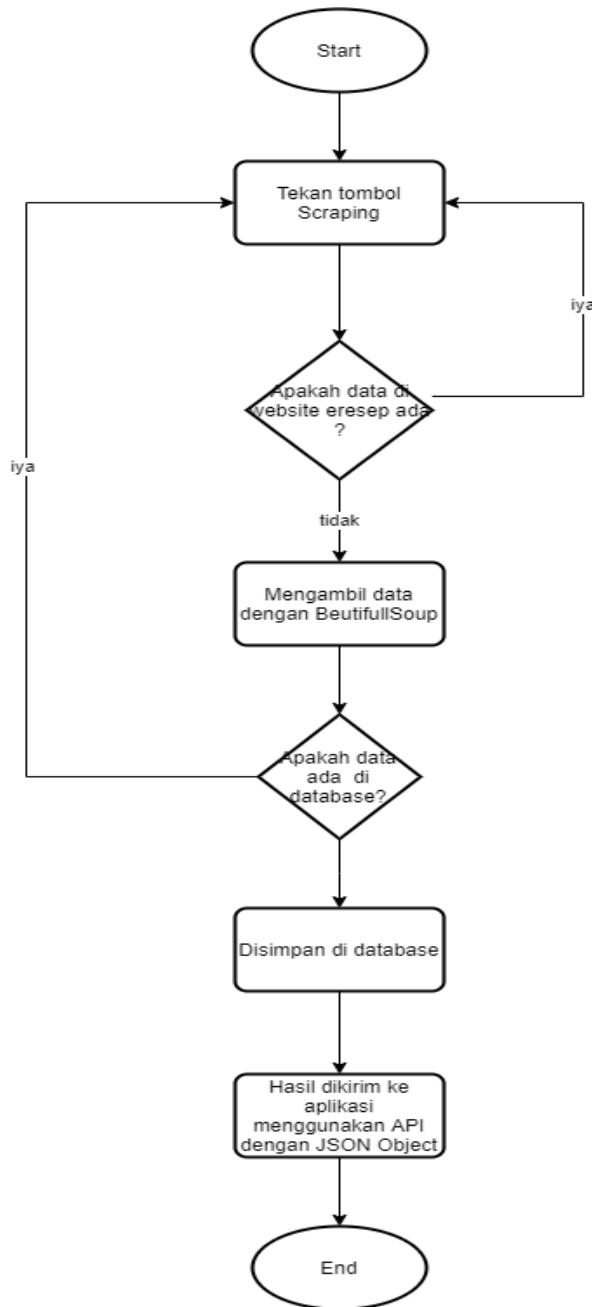
Proses scraping merupakan salah satu proses dalam skripsi ini, data yang didapatkan dari proses scraping ini merupakan menu makanan, bahan makanan dan cara memasak. Perlunya memahami struktur *HTML* dari situs agar data yang diperlukan bisa diambil dengan lancar dan tanpa ada kesalahan.

*Activity diagram* digunakan untuk melihat proses *scraping*, baik di situs eresep atau situs resep lainnya. Pengguna akan menuliskan menu makan yang akan dipilih untuk melakukan *scraping* resep masakan dari situs yang dituju, sehingga sistem akan menjalankan proses scraping. Untuk setiap resep masakan yang terdapat akan diperiksa terlebih dahulu apakah data yang akan diambil sudah terdapat di database. Hal itu dilakukan dengan cara melakukan *query* dengan judul masakan dan author resep masakan. Apabila data belum ada di dalam database, maka proses *scraping* untuk data tersebut diselesaikan dan proses dimasukan ke dalam *database*.



Gambar 3.5 Activity Diagram Proses Scraping Menu Makanan

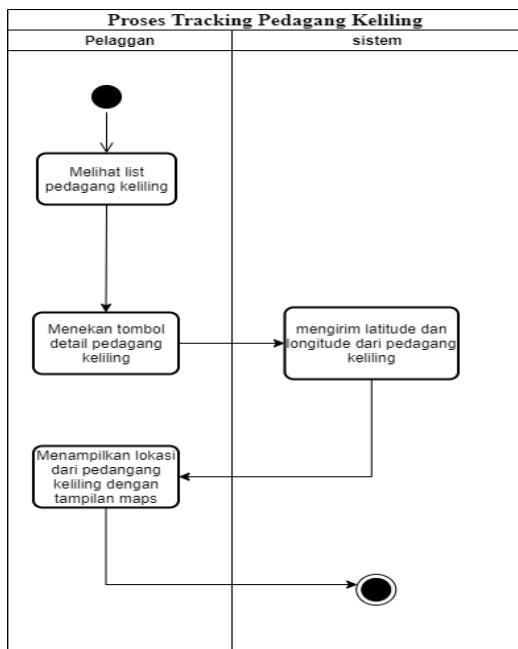
Menggunakan *library python BeautifulSoup* untuk melakukan pengambilan data dari situs eresep.com. *Beautifulsoup* digunakan untuk mengambil data yang ada di situs tersebut. *BeautifulSoup* menggunakan link yang diposting ke dalam *parameter python* yang kemudian digunakan untuk mengakses halaman situs tersebut. Data yang diambil berupa file *JSON* yang didapat di dalam situs tersebut.



Gambar 3.6 Flowchart *Scraping* Makanan

### 3.6.1 Proses Tracking Pedagang Keliling

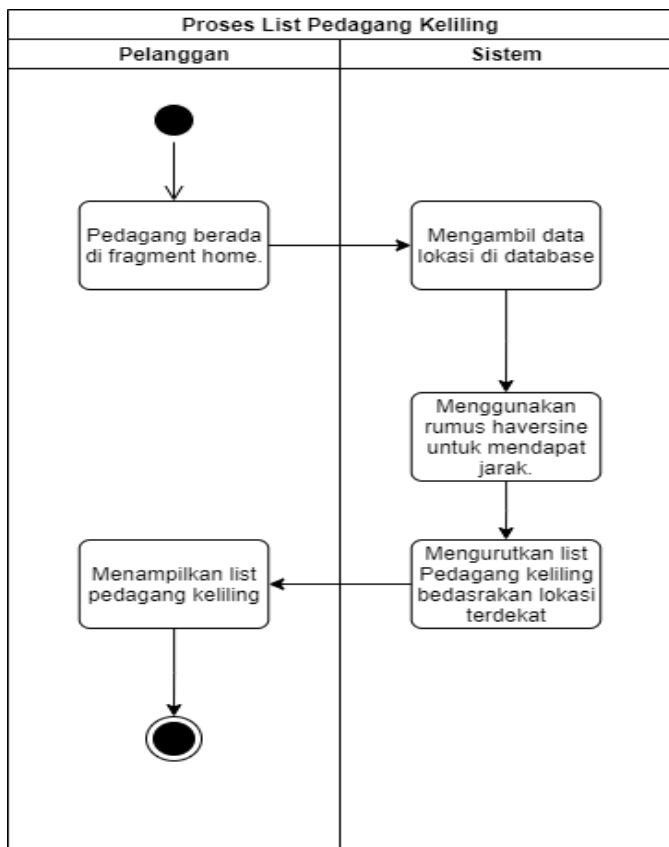
Pada Gambar 3.7 *Activity Diagram Tracking* Pedagang Keliling merupakan activity diagram untuk melakukan pencarian barang atau pedagang keliling. Apabila pengguna masuk ke halaman aplikasi untuk melihat *list* pedagang keliling. Setelah melihat list pedagang keliling maka akan menekan tombol lihat lokasi. Akan masuk kedalam aktivitas detail lokasi pedagang keliling. Di *detail* lokasi pedagang akan terlihat lokasi dari pedagang keliling yang ditampilkan di *google maps*.



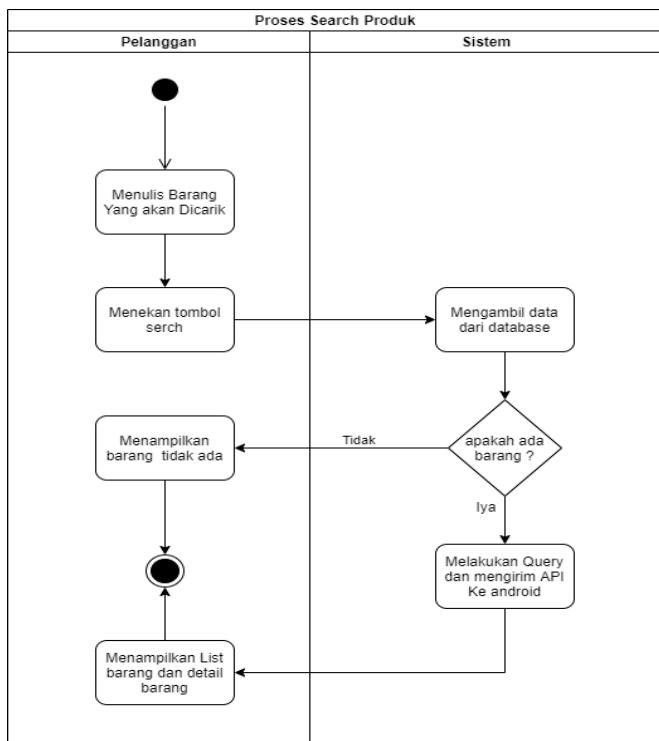
Gambar 3.7 Activity Diagram Tracking Pedagang Keliling

### 3.6.2 Proses List Pedang Keliling dan Search Produk

Menampilkan *list* pedagang keliling berdasarkan jarak terdekat dengan lokasi yang dimiliki saat ini. Melakukan proses *search* dan *filter* sesuai dengan *attribute*. Gambar 3.8 *List* Pedagang Keliling merupakan *activity diagram* untuk menampilkan *list* pedagang keliling berdasarkan lokasi terdekat. Pada Gambar 3.9 *Search* Produk merupakan *activity diagram* untuk melakukan *search* produk berdasarkan *keyword* yang dicari.

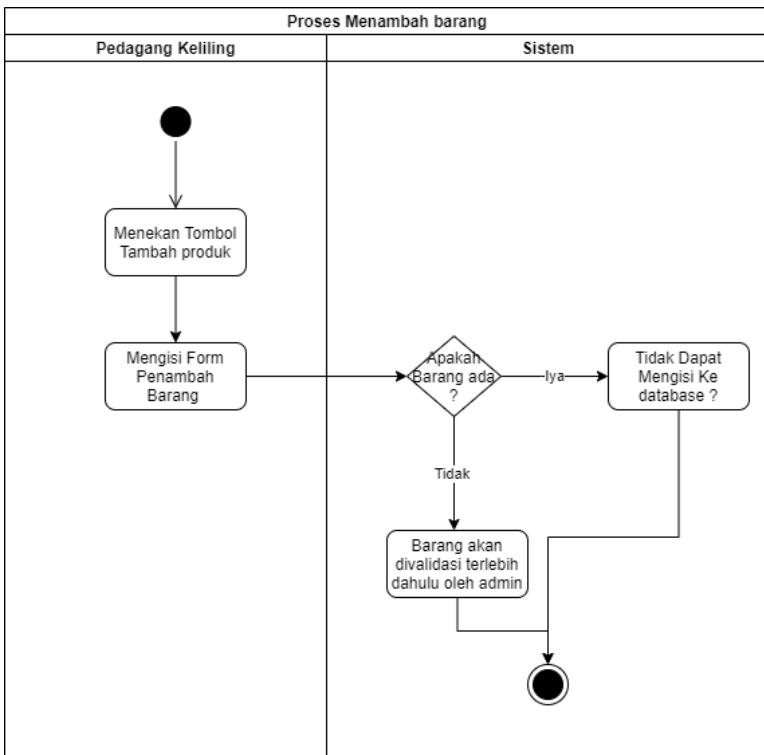


Gambar 3.8 List Pedagang Keliling



Gambar 3.9 Search Produk

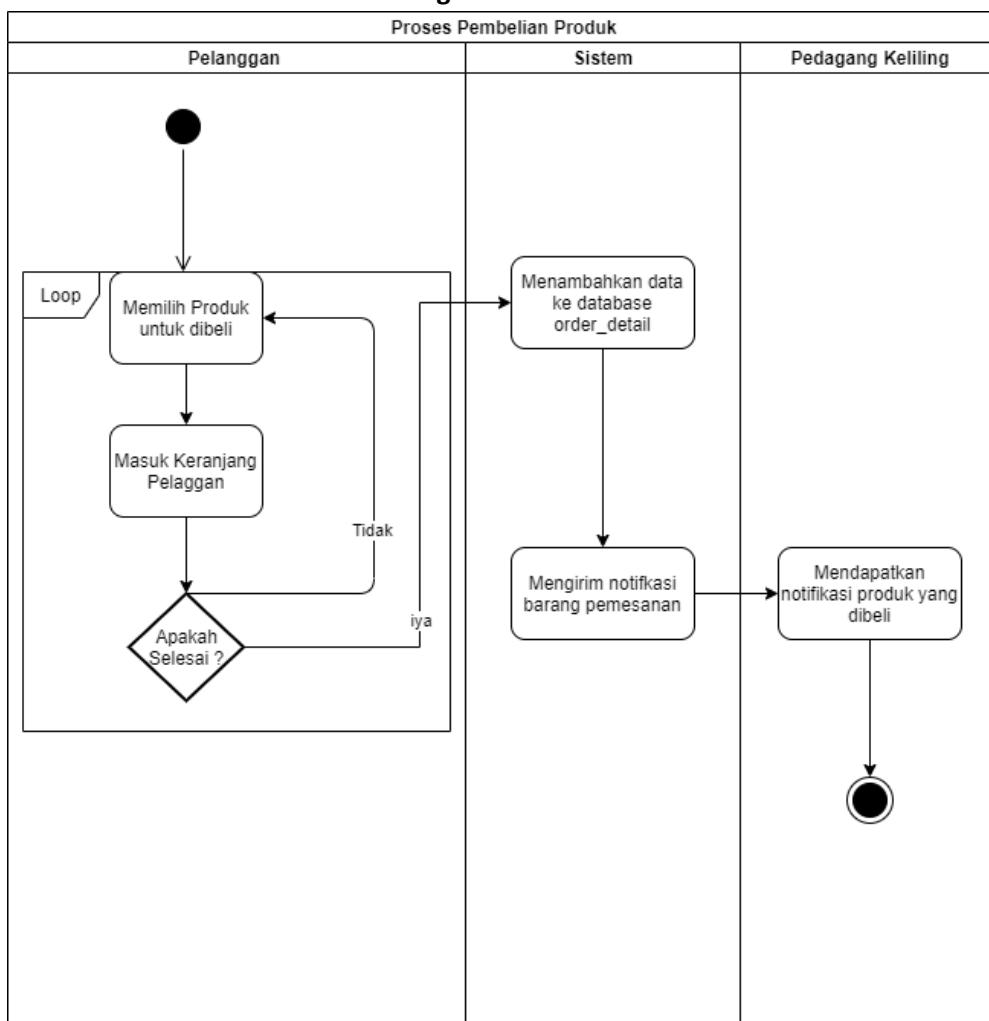
### 3.6.3 Proses Menambah Produk Pedagang Keliling



Gambar 3.10 Menambah Barang

Proses menambah barang merupakan salah satu fitur yang dimiliki oleh pedagang keliling. Gambar 3.10 Menambah Barang merupakan *activity diagram* untuk proses menambah barang . Menambahkan barang yang akan dibawakan atau dijual oleh pedagang keliling .

### 3.6.4 Proses Pembelian Barang



Gambar 3.11 Proses Pembelian Produk

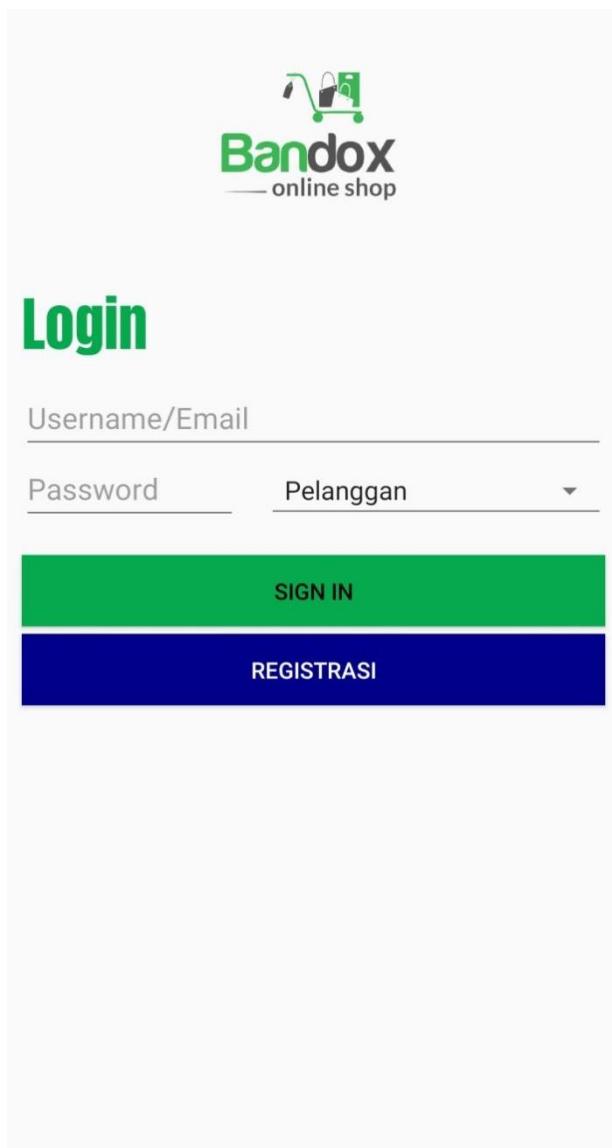
Pada Gambar 3.11 Proses Pembelian Produk proses pembelian barang merupakan fitur pelanggan untuk melakukan pembelian barang. Pertama pelanggan akan menginput barang – barang kedalam keranjang belanjaan dan lalu barang akan diantar oleh pedagang keliling.

### 3.7 Desain Interface Pelanggan

Membentuk tampilan yang dibutuhkan oleh kebutuhan *user* pelanggan. Tampilan pertama adalah tampilan Login. Tampilan kedua adalah Registrasi. Tampilan ketiga merupakan tampilan home, aktivitas , menu makanan ,profile dan history user.

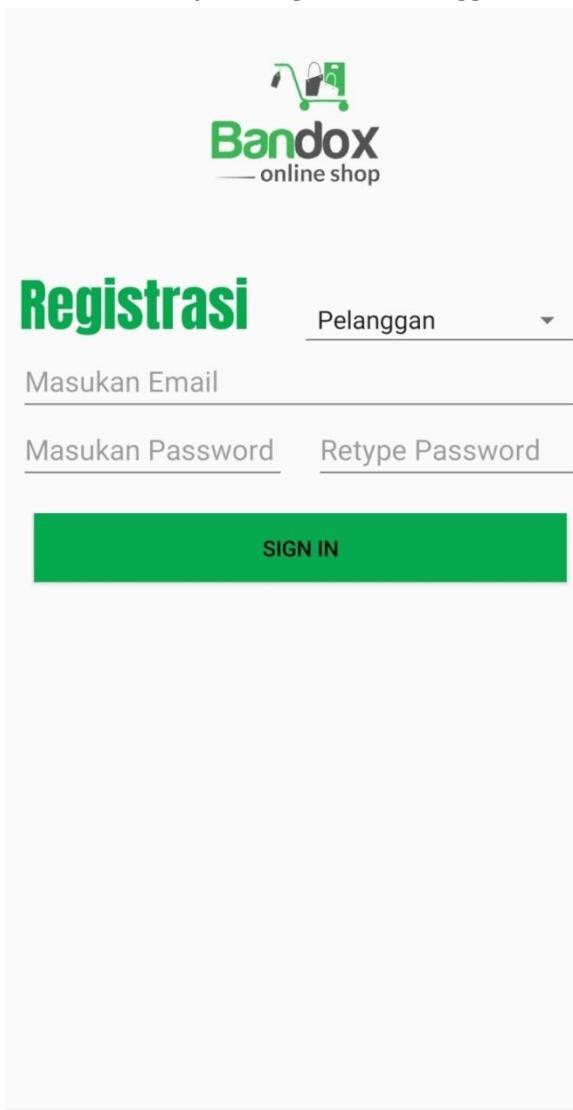
### **3.7.1 Tampilan Login Pelanggan**

Tampilan desain login dengan tema berwarna hijau. Melakukan pengecekan *user* pelanggan melalui API.



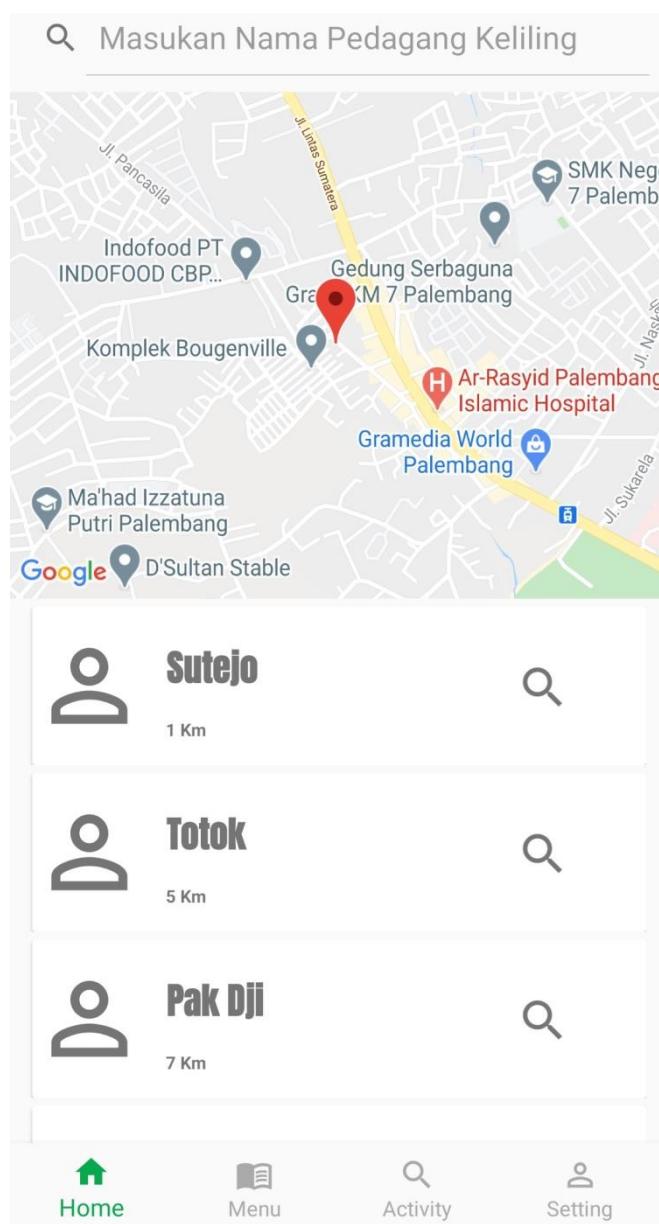
Gambar 3.12 Tampilan Login Pelanggan

**3.7.2 Tampilan Registrasi Pelanggan**



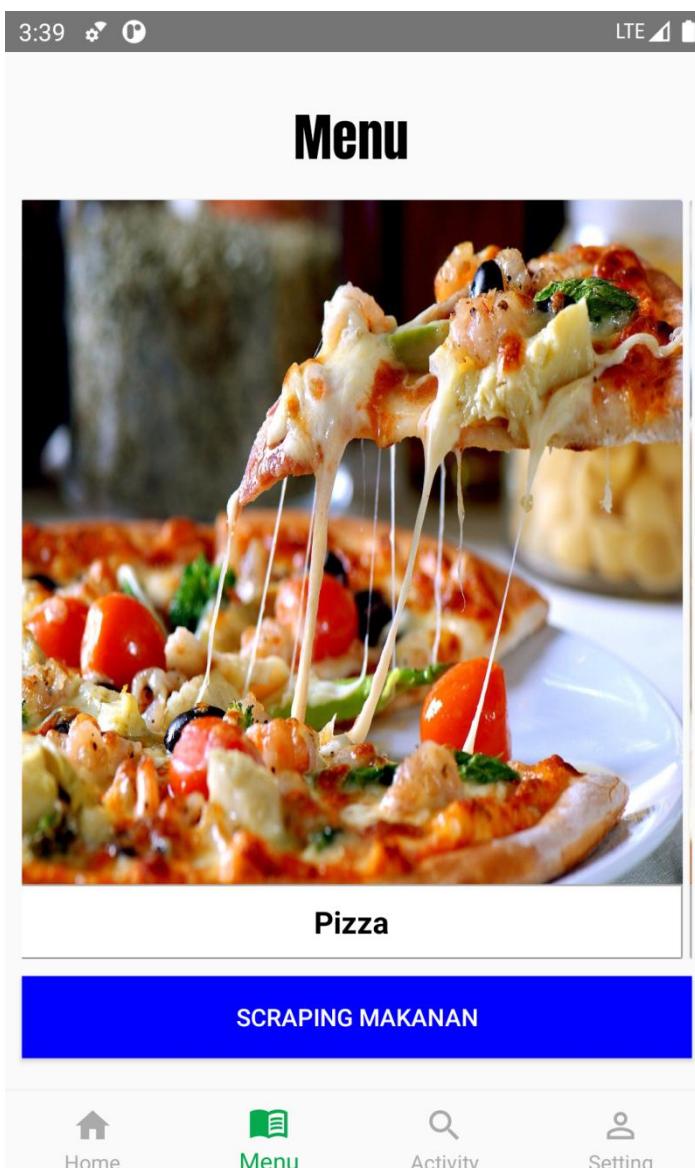
Gambar 3.13 Tampilan Registrasi Pelanggan

### 3.7.3 Tampilan Home Pelanggan



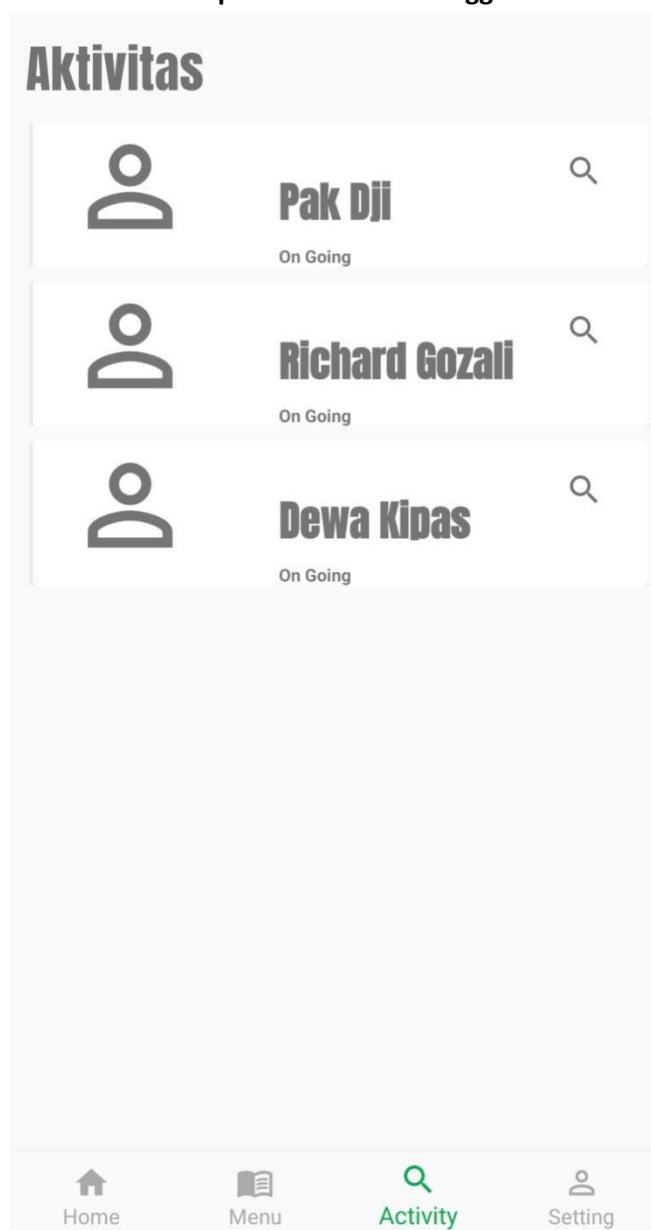
Gambar 3.14 Tampilan Home Pelanggan

### 3.7.4 Tampilan Menu Makanan Pelanggan



Gambar 3.15 Tampilan Menu Makanan

### 3.7.5 Tampilan Aktivitas Pelanggan



Gambar 3.16 Tampilan Menu Aktivitas Pelanggan

### 3.7.6 Tampilan Detail Pedagang



Bumbu

Bawang Putih



0

Rp 1000



Bawang Merah



0

Rp 5000



Gambar 3.17 Tampilan Detail Pelanggan



Bumbu

Bawang Putih



0

Rp 1000



Bawang Merah



0

Rp 5000



Ikan

Cabe Merah



0

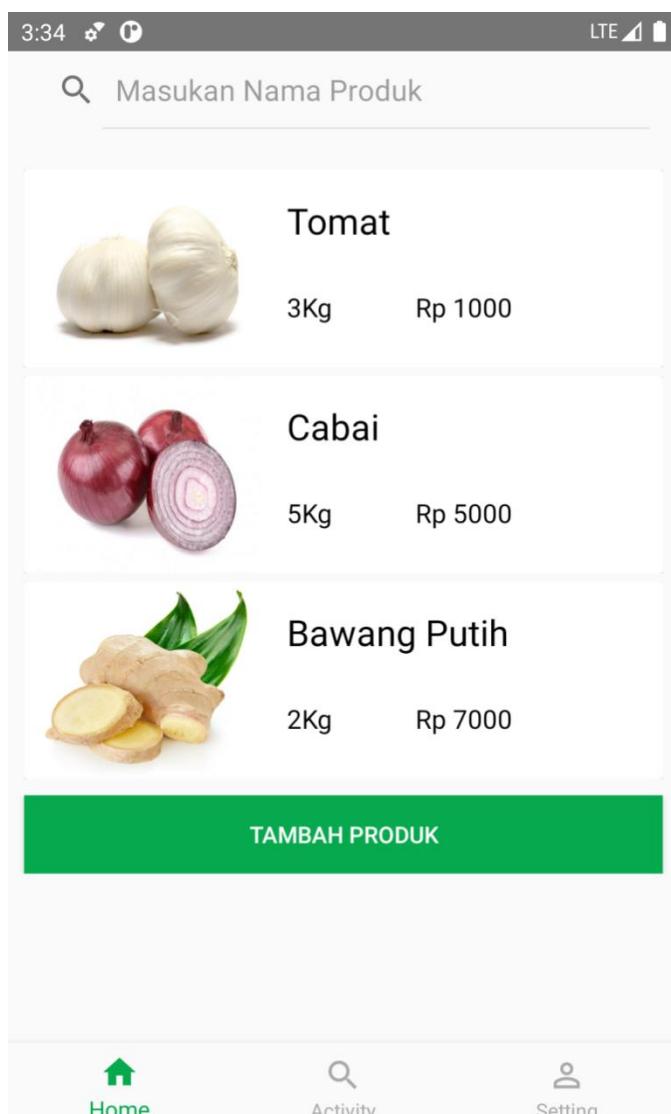
Rp 7000



Gambar 3.18 Tampilan Detail Pelanggan

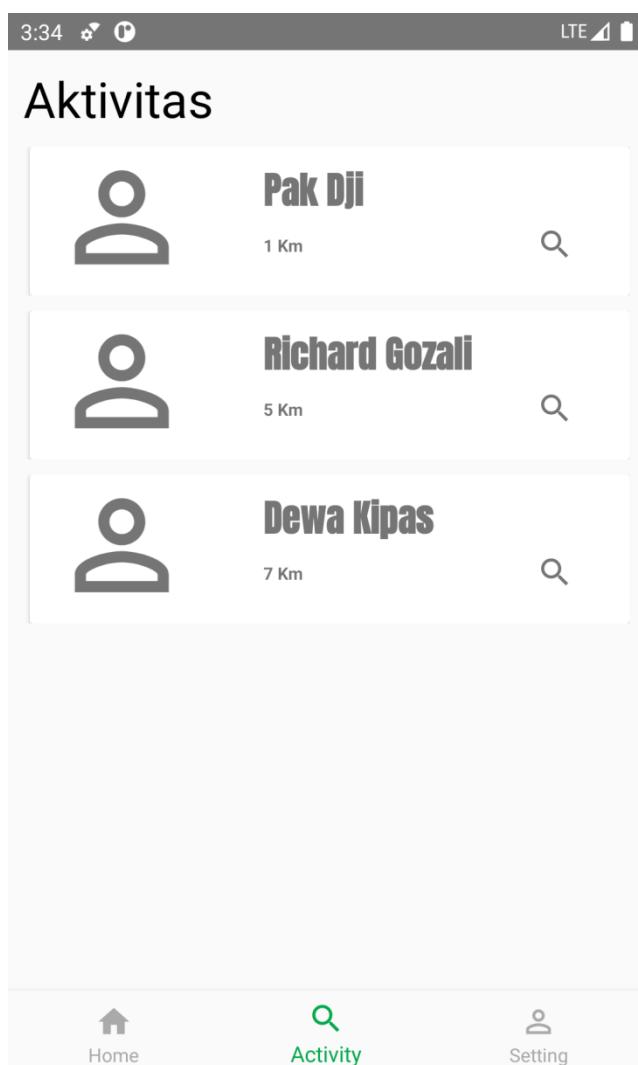
### 3.8 Desain Interface Pedagang Keliling

#### 3.8.1 Desain Home Pedagang Keliling



Gambar 3.19 Tampilan Home Pedagang Keliling

### 3.8.2 Desain Aktivitas Pedagang Keliling



Gambar 3.20 Tampilan Aktivitas Pedagang Keliling

## BAB 4. IMPLEMENTASI SISTEM

Pada bab ini akan dibahas tentang implementasi sistem sesuai dengan analisa dan desain sistem. Implementasi sistem meliputi implementasi proses *scraping* pada situs *eresep* , proses *tracking pedagang keliling* , proses *list* produk dan pedagang keliling dan proses pembelian barang. Hubungan antara segmentasi program dengan proses yang dilakukan terdapat pada. Selain itu juga terdapat proses di *activity diagram* di bab 3.

Tabel 4.1 Daftar Hubungan *activity diagram* dan segmen program

Proses	Segmen Program	<i>Activity Diagram</i>
Scraping situs <i>eresep.com</i>	Segmen Program 4.2	Gambar 3.5
Proses list pedagang keliling	Segmen Program 4.3	Gambar 3.8
Proses pembelian produk	Segmen Program 4.4	Gambar 3.11
Proses <i>upload</i> produk	Segmen Program 4.5	Gambar 3.10

### 4.1 Proses pembuatan API melalui php dan mysql

Proses ini melakukan *setup* awal untuk pembuatan API dari *php*. Membuat koneksi untuk melakukan *query* ke *database*. Membuat file yang berupa “*database.php*”.

Segmen Program 4.1 *database.php*

```
|?php  
  
$hostname = "localhost";  
$username = "c14170049";  
$password = "TOS2019";  
$database = "c14170049";  
$con = mysqli_connect($hostname, $username, $password, $database);  
// Check connection  
if ($con->connect_error) {  
    die("Connection failed: " . $con->connect_error);  
}  
  
?>
```

### 4.2 Proses Scraping Menu Makanan

Source code untuk proses *scraping* menu makanan pada situs *eresep.com*. Data yang diambil berupa foto makanan, nama makanan, nama pembuat makanan, bahan makanan, dan cara pembuatan makanan. Dari data yang didapat merupakan data yang bertipe *JSON*. Setelah

mendapatkan data yang didapat dari website *eresep.com*. Data tersebut dimasukan kedalam *database*. Pada Segmen Program 4.2 Proses scraping situs *eresep.com* merupakan *source code* proses scraping pada situs *eresep.com*.

Melakukan pemanggilan API kedalam *python script* yang kemudian akan disimpan kedalam *database*. Kemudian data akan dikirim melalui API yang bertipe file *JSON*. Kemudian di aplikasi android akan mem fetch data tersebut untuk ditampilkan.

Segmen Program 4.2 Proses scraping situs *eresep.com*

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 11 18:37:16 2021

@author: RICHARD
"""

import sys
import json
import requests
from bs4 import BeautifulSoup
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="c14170049",
    password="T052019",
    database="c14170049"
)

mycursor = mydb.cursor()

url = 'https://www.eresep.com/resep/' +sys.argv[1]

page = requests.get(url)
soup = BeautifulSoup(page.text,'html5lib')

data = json.loads(soup.find('script', type='application/ld+json').text)

nama = data ['author']['name']
nama_makanan = data['name']
foto = data ['image']
resepintruksi = json.dumps(data ['recipeInstructions'])
resepbahan = json.dumps(data ['recipeIngredient'])
idpelanggan = sys.argv[2]
sql = "INSERT INTO `menu_masakan` VALUES (null,%s,%s,%s,%s,%s)"
val = (foto,nama_makanan ,nama, resepbahan, resepintruksi,idpelanggan)
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

Pada segmen program ini data yang didapatkan dari sebuah website eresep.com merupakan data file *JSON*. Setelah *file JSON* didapatkan maka nanti data akan dimasukan ke dalam *database*. Setelah data dimasukan akan diberi pemberitahuan ke dalam aplikasi tersebut(2020, November 27).

```
<?php
    header('Access-Control-Allow-Origin: *');
    header('Content-Type: application/json');

    include "database.php";

    $data = array(
        'msg' => "error"
    );

    $nama_makan = $_GET['nama_makan'];
    $idpelanggan = $_GET['idpelanggan'];
    $string = str_replace(" ", "-", $nama_makan);
    $command = escapeshellcmd("python3 textminning.py $string $idpelanggan");
    $output = shell_exec($command);
    if($output != null){
        $data['msg'] = "success";
        echo json_encode($data);
    }
    else{
        echo json_encode($data);
    }
?>
```

Pada *API* ini melakukan fungsi untuk menjalankan *program python* yang berada di *server*. Setelah fungsi telah dijalankan akan di *encode* status apakah *program* tersebut berhasil atau tidak.

```
class menu_dialog: DialogFragment() {
    private val progressDialog = CustomProgressDialog()
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
        getDialog()!!.getWindow()?.setBackgroundDrawableResource(R.drawable.round_corner);
        val view = inflater.inflate(R.layout.menu_custom_dialog, container, attachToRoot: false)

        return view
    }

    // Initializing a String Array
}

// Tambah Menu Makanan melalui API
private fun tambahmenu(makanan:String){
    val url = "https://tos.petra.ac.id/~c14170049/Skripsi/runpython.php?nama_makan="+namamenu+"&idpelanggan=" + MainActivity.id_user_pelanggan.toString()

    // creating a new variable for our request queue
    val queue = Volley.newRequestQueue(context!!)

    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.GET,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            // on below line we are displaying a success toast message.

            try {
                // on below line we are passing our response
            }
        },
        Response.ErrorListener { error ->
            // on below line we are displaying an error toast message.
        }
    ) {
        override fun getHeaders(): MutableMap<String, String> {
            val headers = HashMap<String, String>
            headers["Content-Type"] = "application/json"
            return headers
        }
    }
    queue.add(request)
}
```

```

try {
    // on below line we are passing our response
    // to json object to extract data from it.
    val respObj = JSONObject(response)

    // below are the strings which we
    // extract from our json object.
    var msg = respObj.get("msg")

    if(msg.toString() == "success"){
        progressDialog.dialog.dismiss()

        val intent = Intent(context, home::class.java)
        startActivity(intent)
        home.idnav=2
    }
    else{
        val builder = AlertDialog.Builder(context!!, R.style.Alertlogin)
        builder.setTitle("PERINGATAN")
        builder.setMessage("MENU MAKANAN TIDAK ADA")
        builder.setPositiveButton( text: "OK", listener: null)
        builder.show()
    }
    progressDialog.dialog.dismiss()
    btn_tambah_menu.isEnabled = true
    dismiss()
} catch (e: JSONException) {
    e.printStackTrace()
}
},
Response.ErrorListener { error -> // method to handle errors.

```

Ini merupakan *program* yang berada di aplikasi android. Memanggil *API* dari *server* setelah data terkirim. Akan terjadi *reload* dan menghasilkan menu makanan yang akan kita cari.

#### 4.3 Proses List Pedagang Keliling

Pada proses list pedagang keliling dan detail pedagang keliling ditampilkan dengan menggunakan *recyclerview* dan data yang diambil menggunakan *JSON* lalu di *fetch* dan ditampilkan ke dalam aplikasi yang ada di android. Disini juga menampilkan detail lokasi dari pedagang keliling yang dituju. Melakukan *request* untuk penggunaan lokasi dan juga menampilkan daerah terdekat dari pedagang keliling. Dengan rumus *haversine* dapat menentukan jarak dari dua koordinat *latitude* dan *longitude*.

Setiap *handphone* atau *device* bergerak secara otomatis koordinat tempat akan di *upload* ke *server* untuk diperbarui terus titik koordinatnya. Di dalam *fragment home* ini juga telah mengambil *token* dari *FCM* untuk *notifikasi*

#### Segmen Program 4.3 Proses List Pedagang Keliling

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    getpedagangkeliling()
    dtNama = arrayListOf()
    dtIdPedagang = arrayListOf()
    dtLokasi = arrayListOf()
    dtLatitude = arrayListOf()
    dtLongitude = arrayListOf()
    Log.d(tag: "Firebase", msg: "token " + FirebaseInstanceId.getInstance().getToken())
    val token = FirebaseInstanceId.getInstance().getToken()
    FirebaseMessaging.getInstance().token.addOnCompleteListener(OnCompleteListener { task ->
        if (!task.isSuccessful) {
            Log.w(TAG, msg: "Fetching FCM registration token failed", task.exception)
            return@OnCompleteListener
        }

        // Get new FCM registration token
        val token = task.result
        updateToken(token.toString())
    })
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    // fungsi manggil MAPS
    /*val mapFragment = childFragmentManager.findFragmentById(R.id.map) as SupportMapFragment?
    mapFragment?.getMapAsync(callback)*/

    if (this.context?.let { it: Context? } != null) {
        ContextCompat.checkSelfPermission(
            it,
            Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(
                this.context as Activity))

```

#### Segmen Program 4.3 Proses List Pedagang Keliling Lanjutan

```

private val callback = OnMapReadyCallback { googleMap ->
    if(location == null){
        val sydney = LatLng(-7.4057672,112.696388)
        latlnguser = sydney
        googleMap.addMarker(
            MarkerOptions()
                .position(sydney)
                .title("Pedagang Keliling")
                .snippet("Pedagang Sayur")
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)))
    }
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(sydney, 15.0f));
    Log.d(tag: "marker" , sydney.toString())
}
else {
    val sydney = location?.latitude?.let { location?.longitude?.let { it1 -> LatLng(it, it1) } }
    googleMap.addMarker(sydney?.let { MarkerOptions().position(it).title("Your's") })
    googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(sydney, 15.0f));
    latlnguser = location?.latitude?.let { location?.longitude?.let { it1 -> LatLng(it, it1) } }
}
for (position in dtnama.indices) {
    val latitudes = dtlatitude[position]
    val longitudes = dtlongitude[position]
    val jarak = dtlokasi[position]
    val jakarta = LatLng(latitudes, longitudes)
    val nama = dtnama[position]
    googleMap.addMarker(
        MarkerOptions()
            .position(jakarta)
            .title(nama)
            .snippet("Berjarak " + jarak.toString() + " Km")
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)))
}
}
}

```

Pada proses ini mengambil lokasi dan kemudian membuat marker yang berada di *Map*. Dan memberikan nama dan juga berapa jarak yang harus ditempuh untuk mencapai titik tersebut.

### Segmen Program 4.3 Proses List Pedagang Keliling Lanjutan

```
val url = "https://tos.petra.ac.id/~c14170049/Skripsi/getpedagangkeliling.php"

// creating a new variable for our request queue
val queue = Volley.newRequestQueue(context!!)

// on below line we are calling a string
// request method to post the data to our API
// in this we are calling a post method.

val request: StringRequest = object : StringRequest(
    Request.Method.GET,
    url,
    Response.Listener { response ->
        // inside on response method we are
        // hiding our progress bar
        // and setting data to edit text as empty
        // on below line we are displaying a success toast message.

        try {
            // on below line we are passing our response
            // to json object to extract data from it.
            val respObj = JSONObject(response)

            // below are the strings which we
            // extract from our json object.
            Log.d( tag: "response", respObj.toString())
            var msg = respObj.get("msg")

            if(msg.toString() == "success"){
                for (i in 0 until respObj.length() - 1) {
                    val pedagang = respObj.getJSONObject(i.toString())
                    val id_pedagang = pedagang.get("id_pedagang")
                    val telepon = pedagang.get("telepon")
                    val nama_lengkap = pedagang.get("nama_lengkap")
                    val latitude = pedagang.get("latitude")
                    val longitude = pedagang.get("longitude")
                    val r_latitude = pedagang.get("region_latitude")
                    val r_longitude = pedagang.get("region_longitude")
                    Log.d( tag: "nama" , nama_lengkap.toString())
                }
            }
        }
    }
)
```

Pada proses melakukan *request API* yang berada di *server* mengambil data yang berasal di *server* kemudian mengambil data berupa lokasi, nama lengkap, telepon dan id pedagang.

### Segmen Program 4.3 Proses List Pedagang Keliling Lanjutan

```
val telepon = pedagang.get("telepon")
val nama_lengkap = pedagang.get("nama_lengkap")
val latitude = pedagang.get("latitude")
val longitude = pedagang.get("longitude")
val r_latitude = pedagang.get("region_latitude")
val r_longitude = pedagang.get("region_longitude")
Log.d("tag", "nama" , nama_lengkap.toString())
if(location != null){
    val jarak = haversine()
    var akhir = jarak.haversine(location!!.latitude, location!!.longitude, latitude.toString().toDouble(), longitude.toString().toDouble())
    dtlokasi.add(akhir.toInt())

    if(akhir.toInt() < 5 ){
        dtnama.add(nama_lengkap.toString())
        dtidpedagang.add(id_pedagang.toString().toInt())
        dtlongitude.add(longitude.toString().toDouble())
        dtlatitude.add(latitude.toString().toDouble())
    }
}
else{
    dtlokasi.add(0)
}

for (position in dtnama.indices) {
    val data = pedagang_keliling(
        dtidpedagang[position],
        dtnama[position],
        dtlokasi[position]
    )
    armenu.add(data)
}
rv_list_pedagang.layoutManager = LinearLayoutManager(context)
adapter = context?.let { adapaterpedagang_keliling(armenu, it) }!!
rv_list_pedagang.adapter = adapter
mapFragment = childFragmentManager.findFragmentById(R.id.map) as SupportMapFragment?
mapFragment?.getMapAsync(callback)
```

Mengambil jarak yang berasal dari *server database* dan kemudian mengolah data lokasi yang berupa *latitude* dan *longitude* menjadi jarak dengan menggunakan rumus *haversine*.

```

fun haversine(lat1: Double, lon1: Double, lat2: Double, lon2: Double): Double {
    val λ1 = toRadians(lat1)
    val λ2 = toRadians(lat2)
    val Δλ = toRadians(lat2 - lat1)
    val Δφ = toRadians(lon2 - lon1)
    return 2 * Companion.R * asin(sqrt(pow(sin(Δλ / 2), 2.0) + pow(sin(Δφ / 2), 2.0) * cos(λ1) * cos(λ2)))
}

```

Merupakan rumus dari *haversine*. R merupakan jari-jari dari bumi yang kemudian melakukan persamaan *haversine* yang menghasilkan hasil berupa jarak yang bertipe *double*.

#### 4.4 Proses Pembelian Produk

Pada proses pembelian produk. Pelanggan akan memilih bahan – bahan makanan yang akan dipilih lalu ditambahkan kedalam keranjang pembelian. Disini menggunakan metode *POST* untuk mengirim data kedalam *API*. Sebelum dapat membeli produk –produk yang dimiliki oleh pedagang keliling akan ditampilkan terlebih dahulu dan pelanggan dapat memilih barang – barang apa saja yang ingin dibeli. Disini mengambil data *JSON* dan menampilkannya.

Setelah pelanggan telah memilih produk yang ingin dibeli. Pelanggan akan menambah jumlah produk yang dinginkan lalu menambahkan ke dalam keranjang. Dengan mengambil id dari produk\_pedagang lalu menambahkan ke dalam *order* tersebut. Setelah itu menghasilkan keranjang pada setiap pelanggan.

## Segmen Program 4.4 Pembelian Produk

```
class detail_pedagang : AppCompatActivity(), OnMapReadyCallback {
    fun reload(){
        getjenisproduk(id_pedagang.toString())
    }
    companion object{
        var id_pedagang = 0
        fun refresh(){
        }
    }

    private lateinit var adapter: adapterdetail_pedagang
    private lateinit var dtnama: ArrayList<String>
    private lateinit var dtfoto : ArrayList<String>
    private lateinit var dtharga : ArrayList<Int>
    private lateinit var dtiumlah : ArrayList<Int>
    private lateinit var dtkategori : ArrayList<String>
    private lateinit var dtlatitude : ArrayList<Double>
    private lateinit var dtlongitude : ArrayList<Double>
    private lateinit var dtidproduk :ArrayList<Int>
    private lateinit var namapedagang : String
    private lateinit var idorder:String
    private lateinit var dt_tipe :ArrayList<Int>

    private var armenu = arrayListOf<produk_detail>()
    private var hasjenis: HashMap<Int, String> = HashMap<Int, String>()

    private fun getjenisproduk(id_pedagang: String){
        val url = "https://tos.petra.ac.id/~c14170049/Skripsi/getjenis.php"

        // creating a new variable for our request queue
        val queue = Volley.newRequestQueue( context: this)
        // on below line we are calling a string
        // request method to post the data to our API
        // in this we are calling a post method.

        val request: StringRequest = object : StringRequest(
            Request.Method.POST,
            url,
            Response.Listener { response ->
```

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
    Response.Listener { response ->
        // inside on response method we are
        // hiding our progress bar
        // and setting data to edit text as empty
        // on below line we are displaying a success toast message.

        try {
            // on below line we are passing our response
            // to json object to extract data from it.
            val respObj = JSONObject(response)

            // below are the strings which we
            // extract from our json object.
            var msg = respObj.get("msg")

            if(msg.toString() == "error"){
                }
            else{

                for (i in 0 until respObj.length()-1) {
                    val jenis = respObj.getJSONObject(i.toString())
                    val nama_jenis = jenis.get("nama_jenis")
                    val id_jenis:Int = jenis.get("id_jenis").toString().toInt()
                    val keterangan = jenis.get("keterangan")
                    hasjenis.put( id_jenis,nama_jenis.toString())
                    Log.d( tag: "jenis" , id_jenis.toString())
                }
                getprodukpedagang(id_pedagang)
                getjumlahkeranjang(id_pedagang)
            }
        } catch (e: JSONException) {
            e.printStackTrace()
        }
    },
    Response.ErrorListener { error -> // method to handle errors.

    }) {

}
// below line is to make
// a json object request.
```

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
private fun getlokasipedagang(id_pedagang: String){
    val url = "https://tos.petra.ac.id/~c14170049/Skripsi/getpedagangkelilingbyid.php?id_pedagang="+id_pedagang
    Log.e("tag: "url" , url.toString())
    val queue = Volley.newRequestQueue(context: this)

    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.GET,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            // on below line we are displaying a success toast message.

            try {
                // on below line we are passing our response
                // to json object to extract data from it.
                val respObj = JSONObject(response)

                // below are the strings which we
                // extract from our json object.
                var msg = respObj.get("msg")

                if(msg.toString() == "success") {
                    for (i in 0 until respObj.length() - 1) {
                        val pedagang = respObj.getJSONObject(i.toString())
                        val nama = pedagang.get("nama_lengkap")
                        val latitude = pedagang.get("latitude").toString()
                        val longitude = pedagang.get("longitude").toString()
                        namapedagang = nama.toString()
                        dtlatitude.add(latitude.toDouble())
                        dtlongitude.add(longitude.toDouble())
                    }
                    val mapFragment = supportFragmentManager.findFragmentById(R.id.map) as? SupportMapFragment
                    mapFragment?.getMapAsync(this)
                    pb_detail_pedagang.visibility = View.GONE
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        }
    )
}
```

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
private fun getprodukpedagang(id_pedagang:String){
    val url = "https://tos.petra.ac.id/~c14170049/Skripsi/getprodukbyid_pedagang.php?id_pedagang="+id_pedagang
    Log.e("tag: " + "url" , url.toString())
    val queue = Volley.newRequestQueue( context: this)

    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.GET,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            // on below line we are displaying a success toast message.

            try {
                // on below line we are passing our response
                // to json object to extract data from it.
                val respObj = JSONObject(response)

                // below are the strings which we
                // extract from our json object.
                var msg = respObj.get("msg")
                var tipe = 2
                var temp = ""
                if(msg.toString() == "success") {
                    for (i in 0 until respObj.length() - 1) {
                        val produk = respObj.getJSONObject(i.toString())
                        val nama_produk = produk.get("nama_produk")
                        val id_produk = produk.get("id_produk_pedagang")
                        val foto = produk.get("img_produk")
                        val jumlah = produk.get("jumlah")
                        val harga = produk.get("harga").toString().toInt()
                        val unit = produk.get("unit").toString()
                        val jenis = produk.get("id_jenis").toString().toInt()
                        val key = hasjenis.get(jenis)
                        if(temp == key!!){
                            tipe = 1
                        }
                    }
                }
            }
        }
    )
}
```

Pada proses ini data telah di *fetch* dari *API* kemudian akan ditampilkan barang apa saja yang telah dimiliki oleh pedagang keliling. Kemudian dengan mengelola data dengan mengurutkan berdasarkan jenis kemudian akan ditampilkan barang sesuai jenisnya dan menghasilkan tampilan yang lebih baik.

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
fun getjumlahkeranjang(id_pedagang:String){

    val url = "https://tos.petra.ac.id/~c14170049/Skripsi/getjumlahkeranjang.php"
    val queue = Volley.newRequestQueue( context this)
    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.POST,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            // on below line we are displaying a success toast message.

            try {
                // on below line we are passing our response
                // to json object to extract data from it.
                val respObj = JSONObject(response)
                // below are the strings which we
                // extract from our json object.
                var msg = respObj.get("msg")
                if(msg == "success"){
                    btn_keranjang.setText(respObj.get("count").toString())
                    idorder = respObj.get("id_order").toString()
                }
            } catch (e: JSONException) {
                e.printStackTrace()
            }
        },
        Response.ErrorListener { error -> // method to handle errors.

    ) {
        override fun getParams(): Map<String, String> {
            // below line we are creating a map for
            // storing our values in key and value pair.
            val params: MutableMap<String, String> =
                HashMap()
    
```

Pada proses ini mengambil *id\_order* dari sebuah pembelian dan jumlah berapa banyak jumlah produk yang telah ingin dibeli kemudian dari hasil tersebut akan ditampilkan di tampilan *button* keranjang.

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    idorder = "0"
    setContentView(R.layout.activity_detail_pedagang)
    dtlongitude = arrayListOf()
    dtlatitude = arrayListOf()
    dtfoto = arrayListOf()
    dtjumlah = arrayListOf()
    dtkategori = arrayListOf()
    dtnama = arrayListOf()
    dtharga = arrayListOf()
    dtidproduk = arrayListOf()
    dt_type= arrayListOf()
    pb_detail_pedagang.visibility = View.VISIBLE
    iv_back.setOnClickListener{ it: View!
        onBackPressed()
    }
    if(id_pedagang == 0){
        id_pedagang = intent.getIntExtra("id_pedagang", defaultValue: 0)
    }

    getjenisproduk(id_pedagang.toString())
    btn_keranjang.setOnClickListener { it: View!
        Log.d("tag: "id_order" , idorder)
        if(idorder == "0"){
            getjenisproduk(id_pedagang.toString())
        }
        else {
            val next = Intent(applicationContext, order_pelanggan::class.java)
            next.putExtra("name: "id_order", idorder)
            next.putExtra("name: "id_pedagang", id_pedagang)
            Log.e("tag: "id_pedagang", id_pedagang.toString())
            startActivity(next)
        }
    }
}
```

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
private fun tambahorder(id_pelanggan:String , id_pedagang:String,idproduk:String,harga:String,jumlah:String){

    val url ="https://tos.petra.ac.id/~c14170049/Skripsi/insertorder.php"
    // creating a new variable for our request queue
    val queue = Volley.newRequestQueue(conten)
    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.POST,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            // on below line we are displaying a success toast message.

            try {
                // on below line we are passing our response
                // to json object to extract data from it.
                val respObj = JSONObject(response)

                // below are the strings which we
                // extract from our json object.
                Log.d( tag: "response" , respObj.toString())
                AlertDialog.Builder(conten)
                    // Judul
                    .setTitle("Perhatian")
                    // Pesan yang di tampilkan
                    .setMessage("Data Telah Ditambah")
                    .setPositiveButton( text: "OK" , DialogInterface.OnClickListener { dialogInterface, i -> })
                    .show()
                val pIntent2 = Intent (conten,detail_pedagang::class.java)
                pIntent2.putExtra( name: "id_pedagang",id_pedagang)
                pIntent2.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
                conten.startActivity(pIntent2)

            } catch (e: JSONException) {
                e.printStackTrace()
            }
        },
        Response.ErrorListener { error -> // method to handle errors.
    }
},
```

Proses ini menambah data produk yang ingin dibeli dan menyimpan datanya kedalam *database*. Kemudian akan dilakukan *reload* untuk tampilan ulangnya.

## Segmen Program 4.4 Pembelian Produk Lanjutan

```
<?php

header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json');

include "database.php";

$data = array(
    'msg' => "error"
);

$id_pelanggan = $_POST['id_pelanggan'];
$id_pedagang = $_POST['id_pedagang'];
$id_produk_pedagang = $_POST['id_produk_pedagang'];
$harga = $_POST['harga'];
$jumlah = $_POST['jumlah'];
$subtotal = $harga * $jumlah;

$sql = "SELECT * FROM `order` WHERE status = 0 and id_pedagang = $id_pedagang and id_pelanggan = $id_pelanggan";
$res = mysqli_query($con,$sql);

if(mysqli_num_rows ( $res ) > 0){
    $data['msg'] = "Data sudah ada";

    $id_order = 0;
    while($row = mysqli_fetch_assoc($res)){
        $id_order = $row['id_order'];
    }
    $sql3 = "INSERT INTO `detail_order`(`id_detail_order`, `id_order`, `id_produk_pedagang`, `harga`, `jumlah`, `sub_total`) VALUES (null,$id_order,$id_produk_pedagang,$harga,$jumlah,$subtotal)";
    $res3 = mysqli_query($con,$sql3);

    echo json_encode($data);
}

else{
    $tanggal = date("Y-m-d");
    $sql12 = "INSERT INTO `order` VALUES (null,$id_pelanggan,$id_pedagang,'$tanggal',0,0,0)";
    $res2 = mysqli_query($con , $sql12);
    if($res2 != null){
        $data['msg'] = "success";

        $id_order = mysqli_insert_id($con);
        $sql3 = "INSERT INTO `detail_order`(`id_detail_order`, `id_order`, `id_produk_pedagang`, `harga`, `jumlah`, `sub_total`) VALUES (null,$id_order,$id_produk_pedagang,$harga,$jumlah,$subtotal)";
        $res3 = mysqli_query($con,$sql3);

        echo json_encode($data);
    }
    else{
        echo json_encode($data);
    }
}
```

Merupakan *query* dari *API* untuk menambahkan produk yang akan dibeli kedalam database. Pertama dengan menquery apakah *id\_order* telah ada jika belum ada maka akan dibuat

terlebih dahulu id\_order kemudian akan menambahkan di detail order yang kemudian akan dijadikan jumlah keranjang.

#### 4.5 Proses Upload Produk

Proses Upload produk merupakan fitur yang dimiliki oleh pedagang keliling untuk mengupload sendiri produk yang dimiliki oleh pedagang keliling. Jika ada produk yang tidak ada dalam data pedagang keliling maka pedagang keliling bisa menambahkannya. Pedagang keliling juga dapat menentukan harga yang ingin dibuat tentukan sendiri.

#### Segmen Program 4.5 Upload Produk

```
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
    getDialog()!!.getWindow()?.setBackgroundDrawableResource(R.drawable.round_corner);
    val view = inflater.inflate(R.layout.custom_dialog, container, attachToRoot: false)

    return view
}

// Initializing a String Array

}

private fun getjenisproduk(){...}
private fun getproduk(){...}
private fun getprodukbyid(id_jenis : String){...}
private fun tambahprodukpedagang(idpedagang:String,idproduk:String,price:String,jumlah:String,unit:String){
    var url = "https://tos.petra.ac.id/~c14170049/Skripsi/insertprodukpedagang.php"

    val queue = Volley.newRequestQueue(context!!)

    // on below line we are calling a string
    // request method to post the data to our API
    // in this we are calling a post method.

    val request: StringRequest = object : StringRequest(
        Request.Method.POST,
        url,
        Response.Listener { response ->
            // inside on response method we are
            // hiding our progress bar
            // and setting data to edit text as empty
            et_harga_dialog.setText("");
            et_jumlah_dialog.setText("");
        }
    )
```

## Segmen Program 4.4 *Upload* Produk Lanjutan

```
// on below line we are displaying a success toast message.

try {
    // on below line we are passing our response
    // to json object to extract data from it.
    val respObj = JSONObject(response)

    // below are the strings which we
    // extract from our json object.
    Log.d( tag: "response", respObj.toString())
    var msg = respObj.get("msg")

    if(msg.toString() == "Already INSERT to Database"){
        val builder = androidx.appcompat.app.AlertDialog.Builder(context!!, R.style.Alertlogin)
        builder.setTitle("Peringatan")
        builder.setMessage("Data Telah Terdaftar")
        builder.setPositiveButton( text: "OK", listener: null)
        builder.show()
        dismiss()
    }
    else{

    }

} catch (e: JSONException) {
    e.printStackTrace()
}
},
Response.ErrorListener { error -> // method to handle errors.
    Toast.makeText(
        context!!,
        text: "Fail to get response = $error",
        Toast.LENGTH_SHORT
    ).show()
}) {
override fun getParams(): Map<String, String> {
    // below line we are creating a map for
    // storing our values in key and value pair.
    val params: MutableMap<String, String> =
        HashMap()
```

Pada Proses ini akan menambahkan produk yang telah diupload ke *server* dan dimasukan kedalam *database*. Proses ini juga mengambil gambar produk menggunakan *Bitmap* yang kemudian di kirim melalui *API* ke *server*.

## Segmen Program 4.4 *Upload* Produk Lanjutan

```
override fun onActivityCreated(savedInstanceState: Bundle?) {
    arrayjenis = arrayListOf()
    super.onActivityCreated(savedInstanceState)
    arrayproduk = arrayListOf()
    arrayunit = arrayOf("PCS", "Kg", "Ikat")
    getjenisproduk()
    getproduk()
    var jenis_spinner : Spinner = spinner_jenis
    jenis_spinner.adapter = ArrayAdapter<String>(context!!, android.R.layout.simple_list_item_1, arrayjenis)

    jenis_spinner.onItemSelectedListener= object : AdapterView.OnItemSelectedListener{
        override fun onItemSelected(parent:AdapterView<*>, view: View, position: Int, id: Long){
            if(arrayjenis != null) {
                val text: String = parent?.getItemAtPosition(position).toString()
                val key = hasienis.get(text)
                Log.d( tag: "KEY", key.toString())
                getprodukbyid(key.toString())
            }
        }

        override fun onNothingSelected(parent: AdapterView<*>){
            // Another interface callback
        }
    }
}

// Initializing an ArrayAdapter
val spinner: Spinner = spinner_judul_produk
spinner.adapter = ArrayAdapter<String>(context!!, android.R.layout.simple_list_item_1, arrayproduk)
// Set an on item selected listener for spinner object

val spinner2: Spinner = spinner_unit
spinner2.adapter = ArrayAdapter<String>(context!!, android.R.layout.simple_list_item_1, arrayunit)
// Set an on item selected listener for spinner object
```

#### Segmen Program 4.4 *Upload* Produk Lanjutan

```
btn_tambah_produk.setOnClickListener { it: View? ->
    val idproduk = hasproduk.get(spinner_judul_produk.selectedItem.toString())
    val idpedagang = MainActivity.id_user_pedagang
    val price = et_harga_dialog.text.toString()
    val jumlah = et_jumlah_dialog.text.toString()
    val unit = spinner_unit.selectedItem.toString()
    if (idpedagang != null) {
        tambahprodukpedagang(idpedagang, idproduk.toString(), price, jumlah, unit)
    }
}

btn_tambah_barang.setOnClickListener { it: View? ->
    dismiss()
    val intent = Intent (getActivity(), tambah_produk::class.java)
    getActivity()!!.startActivity(intent)
}

override fun onStart() {
    super.onStart()
    val width = (resources.displayMetrics.widthPixels * 0.85).toInt()
    val height = (resources.displayMetrics.heightPixels * 0.40).toInt()
    dialog!!.window?.setLayout(width, ViewGroup.LayoutParams.WRAP_CONTENT)
}
```

Proses ini menambahkan produk yang telah ada didalam database kemudian disimpan kedalam *database* produk milik pedagang dengan itu tidak terjadinya data yang numpuk.

## Segmen Program 4.4 Upload Produk Lanjutan

```
<?php
header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json');

include "database.php";

$data = array(
    'msg' => "error"
);
$nama_produk = $_POST['nama_produk'];
$id_jenis = $_POST['id_jenis'];

$sql = "INSERT INTO `produk` VALUES (null,'','$nama_produk','$id_jenis')";
$res = mysqli_query($con, $sql);

if (!$res){
    $data['msg'] = "Cannot INSERT to Database";
    echo json_encode($data);
}

else{
    $data['msg'] = "Already INSERT to Database";
    $last_id = mysqli_insert_id($con);
    echo json_encode($data);
    $image = $_POST['image'];
    $target_dir = "image";
    $target_dir = $target_dir . "/" . $last_id . ".jpeg";
    file_put_contents($target_dir, base64_decode($image));
    $sql2 = "UPDATE `produk` SET `img_produk`='https://tos.petra.ac.id/~c14170049/Skripsi/$target_dir' WHERE id_produk =$last_id";
    echo $sql2;
    $res2 = mysqli_query($con, $sql2);
    echo json_encode($data);
}

}

?>
```

Proses ini merupakan *query* untuk meng-*upload* data yang dihasilkan *server* dan menamai gambar barang tersebut sesuai dengan *id\_produk*. Dengan hasil ini tidak terjadinya data yang menumpuk yang memungkinkan data ke *replace* atau hilang.

## Bab 5 PENGUJIAN SISTEM

Pada bab ini akan dijelaskan tentang pengujian terhadap aplikasi yang telah dibuat. Pengujian sistem dilakukan agar dapat memastikan apakah sistem telah berjalan dengan lancar dan baik. Baik proses *scraping* , proses pembelian produk , proses *upload* produk dan juga proses untuk melihat posisi pedagang keliling maupun pelanggan dan juga fitur fitur tambahan lainnya. Pengujian sistem dilakukan dengan cara menjalankan aplikasi secara keseluruhan.

### 5.1 Pengujian Data Produk Pedagang Keliling

Pengujian data untuk mengetahui apa saja barang bawaan yang telah dibawa oleh pedagang keliling dan juga rata – rata harga produk yang dibawa oleh pedagang keliling tersebut. Telah dilakukan survey terhadap lima pedagang keliling dan telah dihasilkan barang dagangnya berupa pada gambar Gambar 5.1 *Database Produk Pedagang*. Dengan harga yang rata-rata dari pedagang keliling yang ditemui.

nama_produk	img_produk	jumlah	harga	unit
Bawang Putih	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/1...">https://tos.petra.ac.id/~c14170049/Skripsi/image/1...</a>	10	40000	Kg
Ayam Potong	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/1...">https://tos.petra.ac.id/~c14170049/Skripsi/image/1...</a>	1	30000	PCS
Bawang Merah	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	3	50000	Kg
Lengkuas	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	1	25000	Kg
Tomat	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	10	10000	Kg
Sawi	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	10	3000	Ikat
Daun Bawang	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	10	7000	Ikat
Jahe	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	10	15000	PCS
Ayam Kampung	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	3	60000	PCS
Kacang Panjang	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	10	2000	Ikat
Sawi Putih	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	3	5000	PCS
Kemiri	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	10	2000	Ikat
Kunyit	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/2...">https://tos.petra.ac.id/~c14170049/Skripsi/image/2...</a>	3	15000	Kg
Ketumbar	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	10	1500	PCS
Pala	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	15	2000	PCS
Jinten	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	10	2000	PCS
Wortel	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	10	5000	Ikat
Bayam	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	10	3000	Ikat
Ikan Gurame	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	2	60000	Kg
Ikan Tongkol	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	5	20000	PCS
Ikan Pindang	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	2	5000	PCS
Cumi - Cumi	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/3...">https://tos.petra.ac.id/~c14170049/Skripsi/image/3...</a>	5	10000	PCS
Kangkung	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	3	1500	PCS
Selada Air	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	5	3000	PCS
Kubis	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	6	6000	PCS
Brokoli	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	5	8000	PCS
Kembang Kol	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	5	3000	PCS
Kentang	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	3	16000	Kg
Jagung	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	5	1500	PCS
Semangka	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	2	25000	PCS
Melon	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	5	20000	PCS
Pepaya	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/4...">https://tos.petra.ac.id/~c14170049/Skripsi/image/4...</a>	3	10000	PCS
Sosi Champ isi 3	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	4500	PCS
Ikan Asin	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	2500	PCS
Rawon	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	2000	PCS
Soto	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	2000	PCS
Krengsengan	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	2000	PCS
Bali	<a href="https://tos.petra.ac.id/~c14170049/Skripsi/image/5...">https://tos.petra.ac.id/~c14170049/Skripsi/image/5...</a>	5	2000	PCS

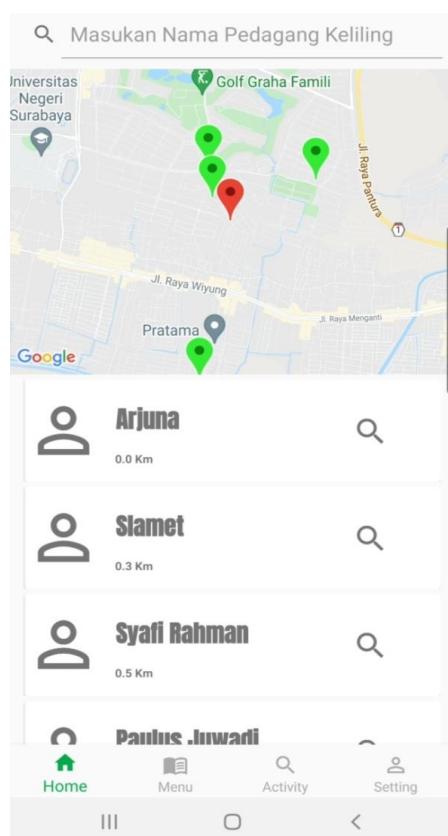
Gambar 5.1 *Database Produk Pedagang*

## 5.2 Pengujian Fitur Pelanggan

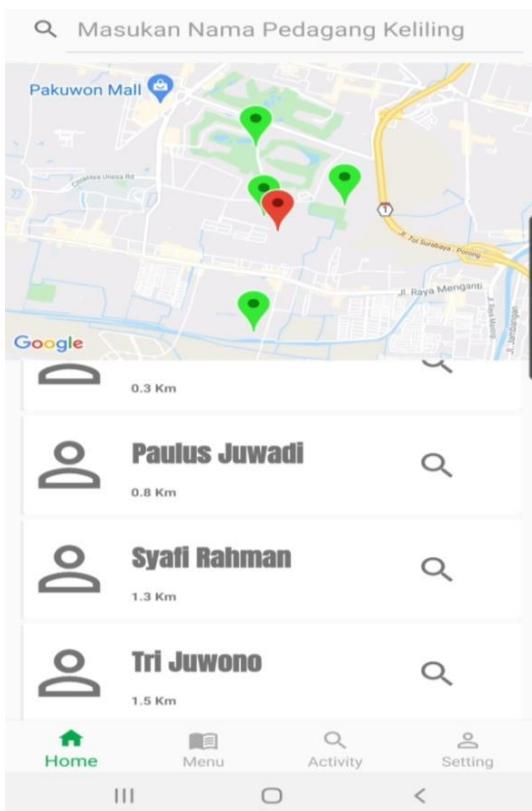
### 5.2.1 Pengujian *Tracking* Pedagang Keliling

Pengujian pada tracking ini dilakukan untuk melihat posisi secara *realtime* dan dapat mengetahui posisi dari pedagang keliling ini secara tepat. Jarak yang didapatkan dengan menggunakan rumus *haversine*. Seperti pada Gambar 5.2 Pengujian *Tracking* Pedagang Keliling pada pengujian ini lokasi pedagang keliling akan selalu diperbarui di *database* dan kemudian akan kembali ditampilkan di aplikasi.

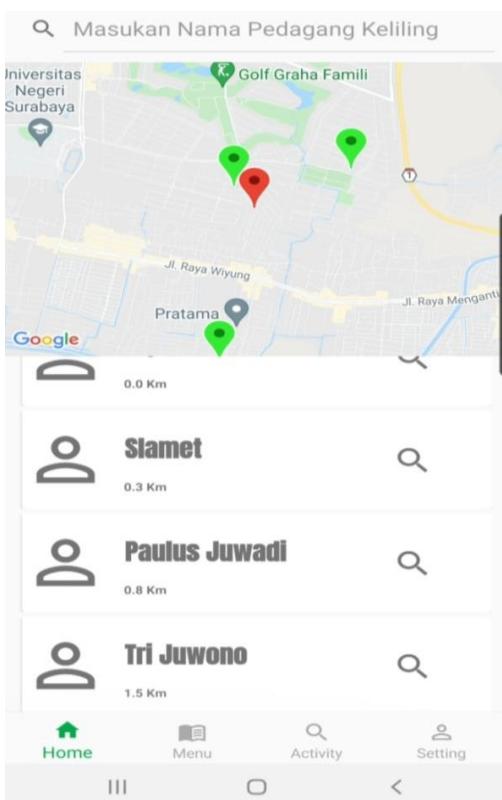
Pada Gambar 5.2 Pengujian *Tracking* Pedagang Keliling ditunjukkan perbedaan lokasi pada Syafi Rahman. Pedagang keliling Syafi Rahman mengalami pergerakan terus menerus pada lokasi 0.5 km. Pada Gambar 5.3 Pengujian *Tracking* Pedagang Keliling Kedua Syafi Rahman mengalami perubahan ke 1.5km. Pada Gambar 5.4 Pengujian *Tracking* Pedagang Keliling Ketiga Syafi Rahman menghilang dari tampilan aplikasi dikarenakan melebihi jarak dari 1.5 km oleh karena itu menghilang dari tampilan aplikasi.



Gambar 5.2 Pengujian *Tracking* Pedagang Keliling



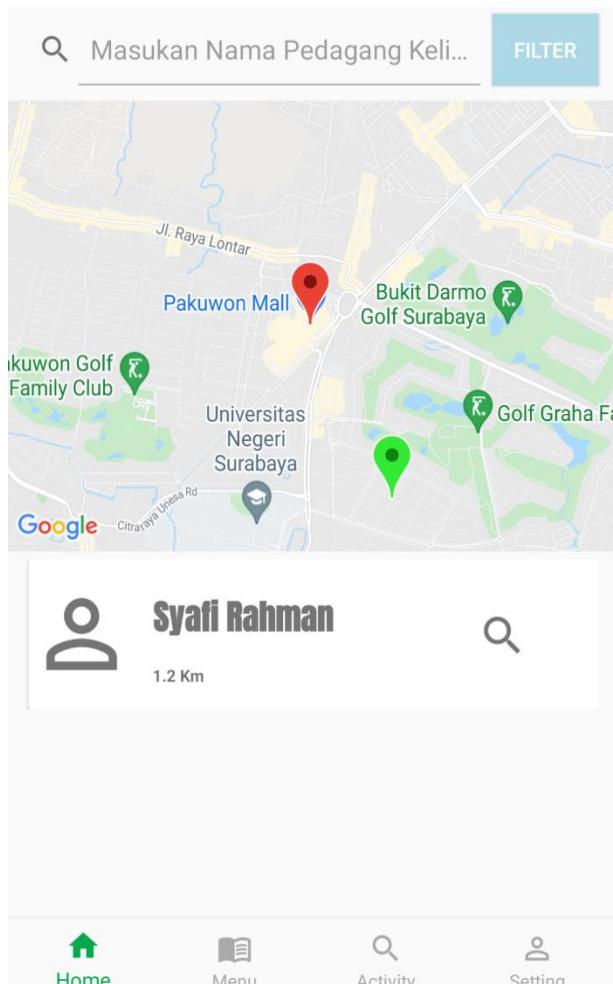
Gambar 5.3 Pengujian *Tracking* Pedagang Keliling Kedua



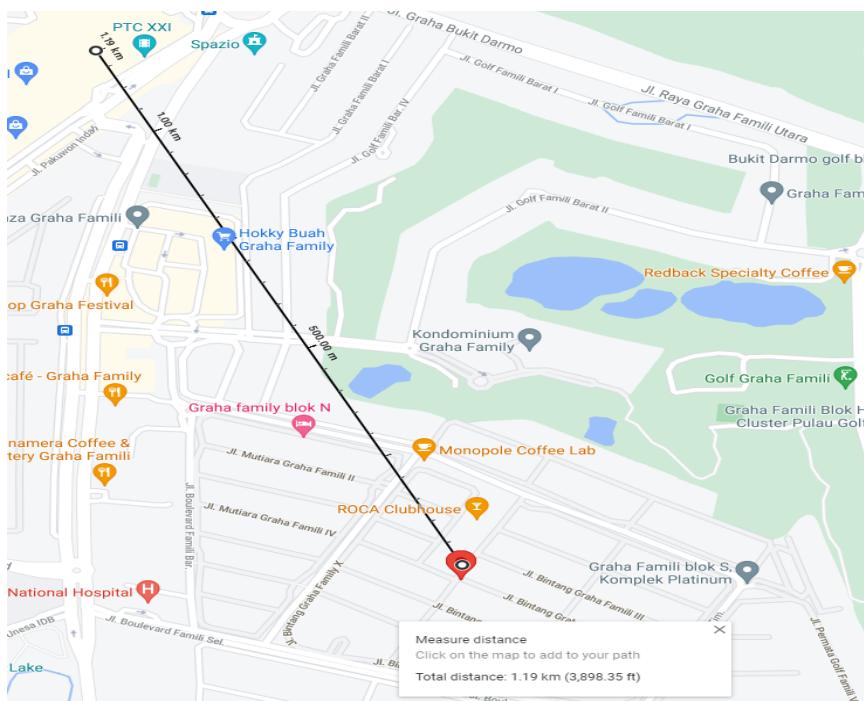
Gambar 5.4 Pengujian *Tracking* Pedagang Keliling Ketiga

Telah dilakukannya pengujian jarak dengan melakukan perbandingan dengan google maps dan *haversine* dengan melakukan perbandingan tersebut. Gambar 5.5 Perbandingan Jarak *Haversine* dan *Google Maps 1*. Hasil pengujian jarak berupa 1.2 km dari lokasi *pakuwon mall*. Setelah dibandingkan dengan hasil yang didapatkan pada Gambar 5.6 Jarak Menggunakan *Google Maps 1*. Dihasilkan jarak 1.19 km dari titik *pakuwon mall*.

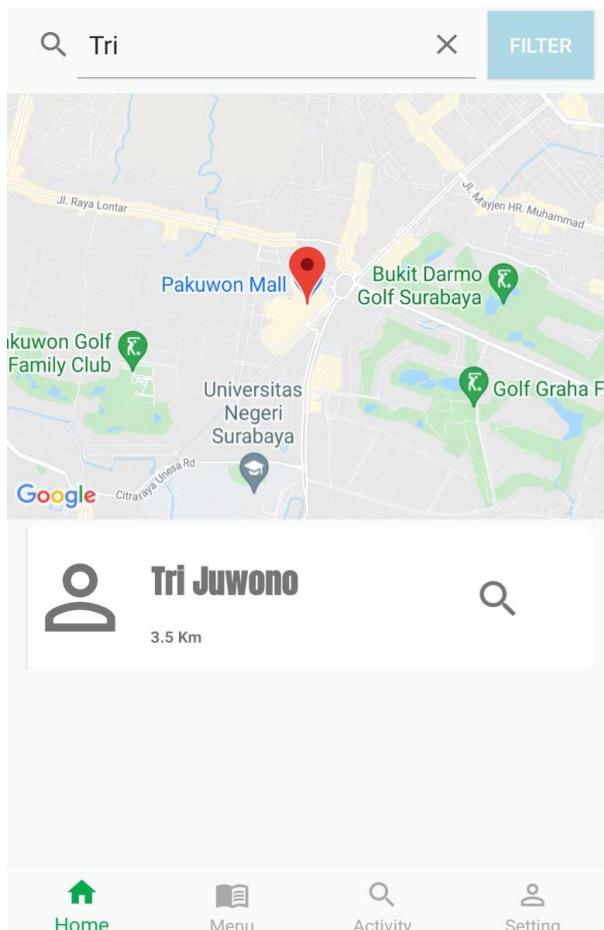
Pada Gambar 5.7 Perbandingan Jarak *Haversine* dan *Google Maps 2* dengan jarak berupa 3.5 km dan dihasilkan jarak dari Gambar 5.8 Jarak Menggunakan *Google Maps 2* adalah 3.52 km dari lokasi *pakuwon mall*. Oleh karena itu hasil pengujian aplikasi ini cukup akurat. Penggunaan *haversine* mudah untuk digunakan ,akurat dan memiliki tingkat error yang rendah.Kekurangan yang dimiliki *haversine* adalah nilai pembulatan bergeser cukup jauh dari hasil yang dihasilkan menggunakan rumus *haversine*.



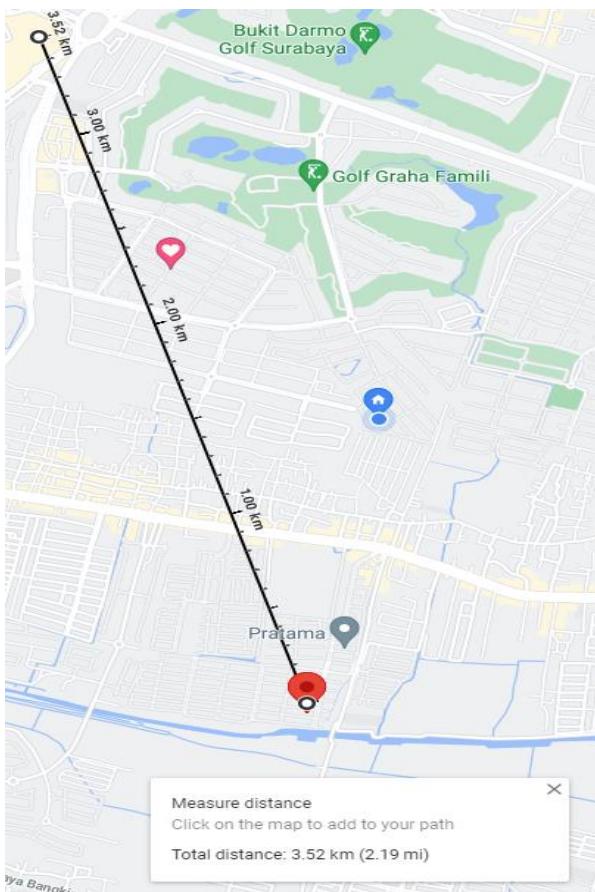
Gambar 5.5 Perbandingan Jarak *Haversine* dan *Google Maps 1*



Gambar 5.6 Jarak Menggunakan Google Maps 1



Gambar 5.7 Perbandingan Jarak Haversine dan Google Maps 2



Gambar 5.8 Jarak Menggunakan *Google Maps* 2

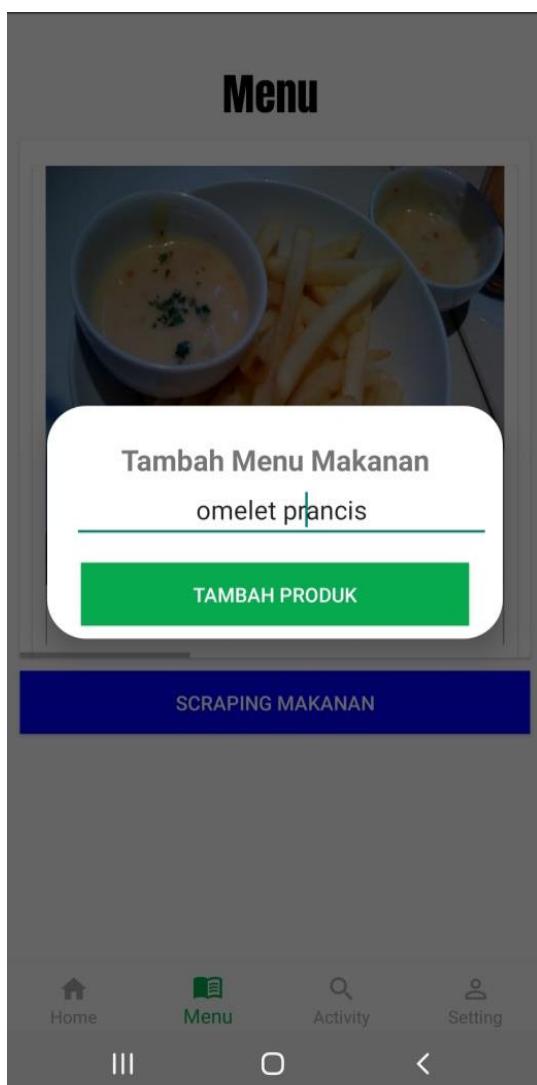
### 5.2.2 Pengujian Scraping Menu Makanan

Pengujian ini bertujuan untuk mengetahui seberapa tepat proses *scraping* data. Selain itu , pengujian dilakukan dengan membandingkan apakah data yang diambil dari situs eresep sudah sama dengan data dari situs itu sendiri. Pengujian Pertama dilakukan dengan menuliskan menu makanan yang dinginkan seperti Gambar 5.9 Pengujian *Scraping* 1 yaitu menu masakan omelet prancis yang kemudian dilakukan *scraping*. Tampilan didalam menu masakan akan bertambah menu baru. Setelah itu *API* akan menyimpan data scraping tersebut kedalam *database* yang kemudian ditampilkan seperti pada Gambar 5.14 *Database* Menu Masakan. Setelah *Scraping* yaitu bahan –bahan makanan berupa 2 telur, 4 putih telur,  $\frac{1}{4}$  cup susu bebas lemak, 1/8 sdt lada,  $\frac{1}{4}$  cangkir daging ham matang yang dipotong dadu , sendok teh bawang Bombay, sendok teh cabai hijau cincang.

Cara memasaknya: Campur dan koncok 5 bahan pertama dari bahan masakan, Tempatkan wajan anti lengket dengan semprotan memasak di api sedang. Tuang adonan telur. Campuran harus segera mengeras di bagian tepinya. Saat telur mengeras, dorong bagian yang sudah

matang ke tengah biarkan telur mentah mengalir di bawahnya. Saat telur mengental dan tidak ada telur cair yang tersisa, bagian atas 1 setengah dengan sisa bahan. Lipat telur dadar menjadi dua. Potong menjadi dua untuk disajikan.

Melakukan pengujian sebanyak lima kali untuk mengetahui berapa lama waktu yang dibutuhkan untuk menambah menu masakan ke dalam *database*. Setelah melakukan pengujian waktu yang dibutuhkan sekitar 4,776 detik. Seperti pada Tabel 5.1 Pengujian Waktu *Scraping* Menu Masakan.



Gambar 5.9 Pengujian *Scraping* 1

## Menu



Omelet Prancis

SCRAPING MAKANAN



Gambar 5.10 Pengujian Scraping 2



### Omelet Prancis

Bahan Makanan :

- 2 telur
- 4 putih telur
- 1/4 cup susu bebas lemak
- 1/8 sdt. garam
- 1/8 sdt. lada
- 1/4 cangkir daging ham matang yang dipotong dadu
- sendok teh. bawang bombay cincang

III O <

Gambar 5.11 Hasil Pengujian Scraping 1

- 1/8 sdt. lada
- 1/4 cangkir daging ham matang yang dipotong dadu
- sendok teh. bawang bombay cincang
- 1 sendok teh. cabai hijau cincang

Cara Masak :

1. Campur dan kocok 5 bahan pertama.
2. Tempatkan wajan anti lengket bisa dilapisi dengan semprotan memasak di atas api sedang. Tuang adonan telur. Campuran harus segera mengeras di tepinya. Saat telur mengeras, dorong bagian yang sudah matang ke tengah, biarkan telur mentah mengalir di bawahnya. Saat telur mengental dan tidak ada telur cair yang tersisa, bagian atas 1 setengah dengan sisa bahan. Lipat telur dadar menjadi dua.
3. Potong menjadi dua untuk disajikan.

III O <

Gambar 5.12 Hasil Pengujian Scarping 2

<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	34	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Pizza	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	33	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Ayam	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	32	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Seafood	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	31	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Kambing	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	30	<a href="https://www.eresep.com/media/">https://www.eresep.com/media/</a>	Takoyaki	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	29	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Seafood	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	28	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Ayam	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	27	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Jawa	eresep

Gambar 5.13 Database Menu Masakan Sebelum Scraping

<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	39	<a href="https://www.eresep.com/media/images/masakan/2021/01/">https://www.eresep.com/media/images/masakan/2021/0... Omelet Prancis</a>	shafa	
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	34	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Pizza	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	33	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Ayam	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	32	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Seafood	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	31	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Kambing	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	30	<a href="https://www.eresep.com/media/">https://www.eresep.com/media/</a>	Takoyaki	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	29	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Seafood	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	28	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Sate Ayam	eresep
<input type="checkbox"/>	 <a href="#">Edit</a>	 <a href="#">Copy</a>	 <a href="#">Delete</a>	27	<a href="https://www.eresep.com/media/images/resep/2020/11/">https://www.eresep.com/media/images/resep/2020/11/...</a>	Nasi Goreng Jawa	eresep

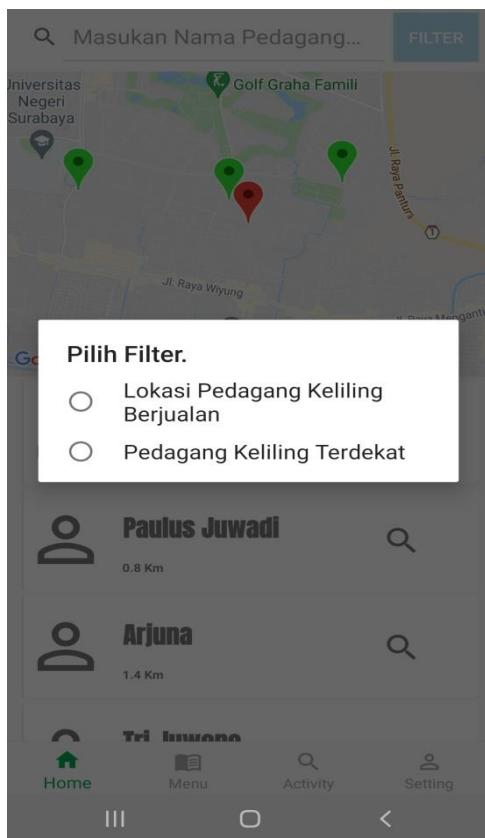
Gambar 5.14 Database Menu Masakan Setelah Scraping

Tabel 5.1 Pengujian Waktu *Scraping* Menu Masakan

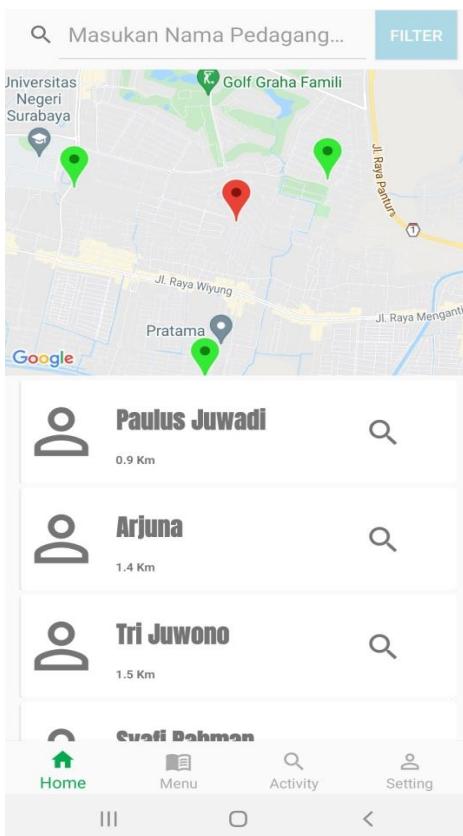
Test 1	6,45detik
Test 2	5,55 detik
Test 3	4,71 detik
Test 4	3,26 detik
Test 5	3,91detik

### 5.2.3 Pengujian Filter Pedagang Keliling *Region*

Proses pengujian untuk fitur pedagang keliling *region*. Menampilkan pedagang keliling yang biasanya berjualan di daerah sekitar kita. Pertama akan memencet tombol filter yang berada di tampilan *home* pelanggan. Kemudian akan muncul *Dialog* yang menunjukan *filter* yang tersedia. Setelah memilih *filter* lokasi pedagang keliling berjualan Seperti pada Gambar 5.15 Hasil Filter Lokasi *Region* Pedagang Keliling kemudian akan muncul *list* pedagang keliling yang berjualan di daerah tersebut dan memunculkan jarak pedagang keliling dengan pelanggan.



Gambar 5.15 Hasil Filter Lokasi *Region* Pedagang Keliling

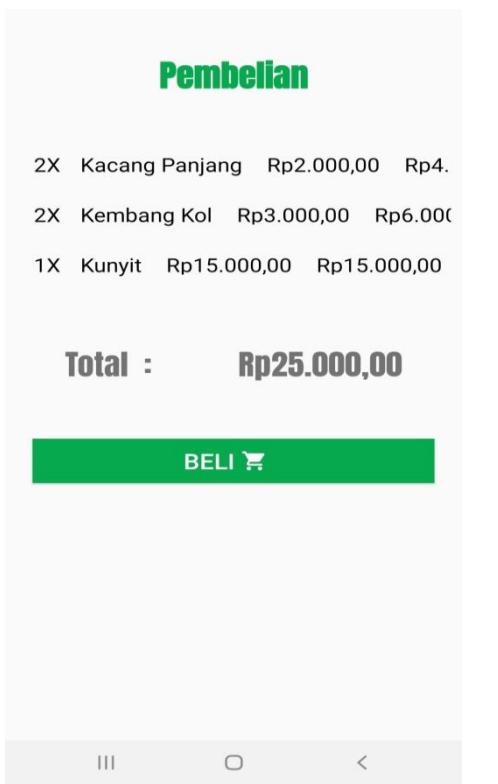


Gambar 5.16 Hasil Filter Lokasi Region Pedagang Keliling

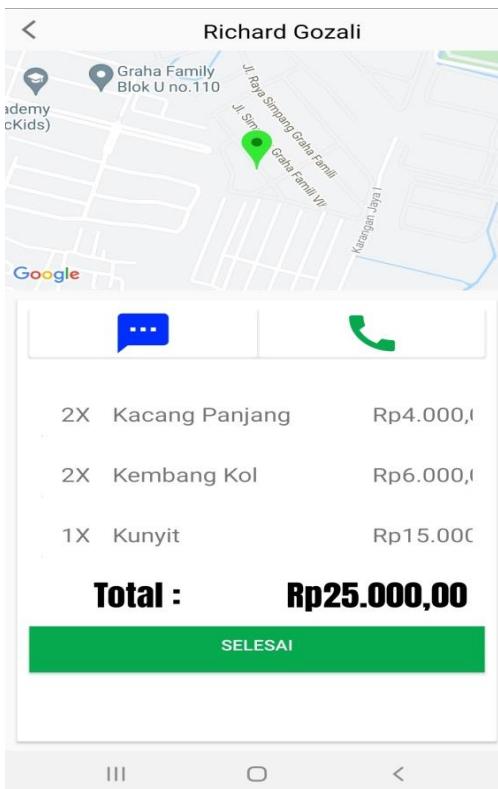
#### 5.2.4 Pengujian Fitur Pembelian Produk

Proses pengujian untuk melakukan pembelian produk terhadap pedagang keliling. Menunjukkan produk apa saja yang telah dibeli dan menunjukkan perkiraan harga produk yang dibeli Seperti pada Gambar 5.17 Pembelian Produk Pedagang Keliling. Pengujian melakukan pembelian produk pedagang berupa 2 kacang panjang dengan harga Rp 2.000,00 per pcs, 2 kembang kol dengan harga Rp 3.000 per pcs dan 1 kunyit dengan harga Rp 15.000,00 per pcs. Setelah menekan tombol pembelian, akan pindah halaman dimana halaman tersebut akan menampilkan lokasi dari pedagang keliling, 2 kacang panjang dengan harga Rp 2.000,00 per pcs, 2 kembang kol dengan harga Rp 3.000 per pcs dan 1 kunyit dengan harga Rp 15.000,00 per pcs dengan total harga Rp 25.000,00.

Setelah barang apa saja yang telah dibeli akan dikirim ke dalam database yang kemudian akan ditampilkan di tampilan pedagang keliling yang seperti pada Gambar 5.18 Data Pembelian Produk di Pedagang Keliling. Pedagang Keliling dapat mengetahui produk pedagang keliling apa saja yang telah dibeli.



Gambar 5.17 Pembelian Produk Pedagang Keliling



Gambar 5.18 Data Pembelian Produk di Pedagang Keliling

## 5.3 Pengujian Fitur Pedagang Keliling

### 5.3.1 Pengujian Upload Produk Pedagang Keliling

Melakukan Pengujian Terhadap waktu yang diperlukan oleh pedagang keliling untuk melakukan *upload* sebuah produk pedagang keliling. Seperti Tabel 5.2 Waktu Upload Produk dilakukanya test sebanyak sepuluh kali untuk menentukan berapa lama waktu yang diperlukan untuk mengirim data yang ke dalam *server*.

Seperti pada Gambar 5.19 Tampilan Upload Produk akan dilakukannya pemilihan produk yang diambil dari *gallery* data ponsel. Dipilih bawang bombay yang memiliki tipe jenis produk yaitu bumbu. Setelah aplikasi dijalankan diperlukan waktu sebesar 3.775 detik untuk dapat melakukan upload produk ke dalam database aplikasi.



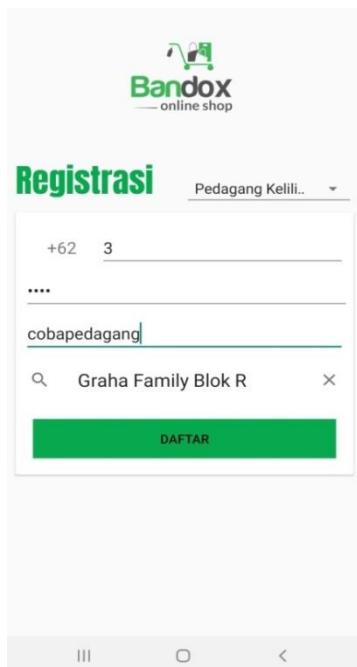
Gambar 5.19 Tampilan Upload Produk

Tabel 5.2 Waktu Upload Produk

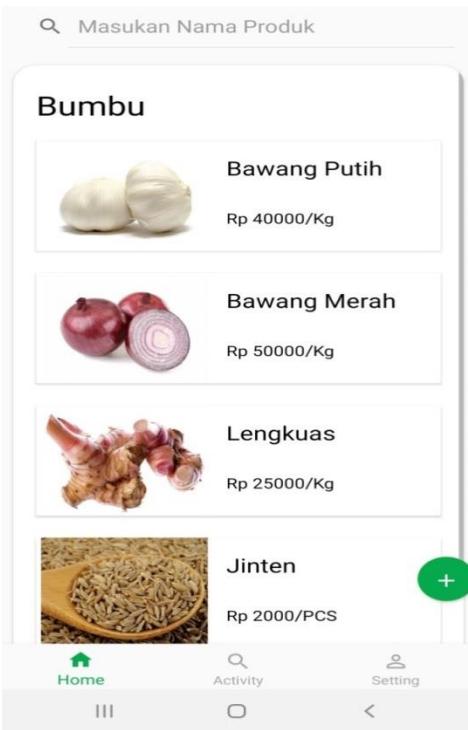
Test 1	3.10 detik
Test 2	2.50 detik
Test 3	4,23 detik
Test 4	5.21 detik
Test 5	3,32 detik
Test 6	4.55 detik
Test 7	3.75 detik
Test 8	4.60 detik
Test 9	5.51 detik
Test 10	3.20 detik

### 5.3.2 Pengujian Fitur Otomatis Tambah Barang

Pada proses pengujian fitur otomatis tambah barang. Barang – barang yang sering dibawa oleh pedagang keliling akan ditambahkan secara otomatis kedalam database produk pedagang keliling dengan ini pedagang tidak perlu untuk menambahkannya sendiri. Pada saat sudah melakukan registrasi pedagang keliling kemudian akan dimasukan data produk pedagang keliling yang telah disiapkan seperti pada Gambar 5.21 Pengujian Tambah Barang Otomatis. Produk – produk pedagang keliling akan disimpan kedalam *database* yang hanya dimiliki oleh pedagang tersebut.



Gambar 5.20 Pengujian Tambah Barang Otomatis 2



Gambar 5.21 Pengujian Tambah Barang Otomatis

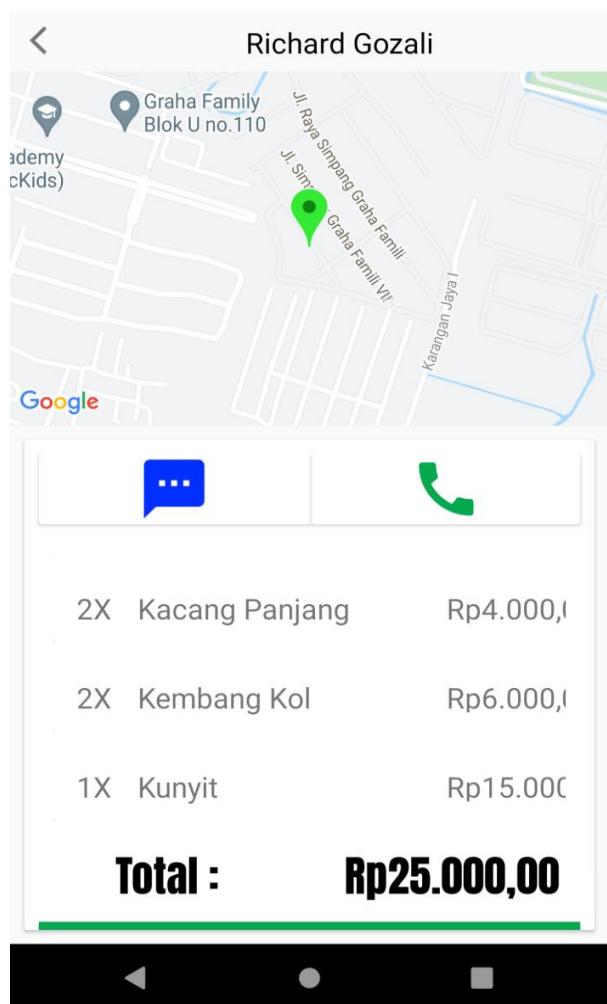
<b>id_pedagang</b>	<b>id_produk</b>	<b>jumlah</b>	<b>harga</b>	<b>unit</b>	<b>id_produk</b>	<b>img_produk</b>	<b>nama_produk</b>
26	1	10	40000	Kg	1	https://tos.petra.ac.id/~c14170049/Skripsi/image/1...	Bawang Putih
26	18	1	30000	PCS	18	https://tos.petra.ac.id/~c14170049/Skripsi/image/1...	Ayam Potong
26	2	3	50000	Kg	2	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Bawang Merah
26	20	1	25000	Kg	20	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Lengkuas
26	21	10	10000	Kg	21	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Tomat
26	22	10	3000	Ikat	22	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Sawi
26	23	10	7000	Ikat	23	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Daun Bawang
26	3	10	15000	PCS	3	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Jahe
26	24	3	60000	PCS	24	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Ayam Kampung
26	25	10	2000	Ikat	25	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Kacang Panjang
26	26	3	5000	PCS	26	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Sawi Putih
26	27	10	2000	Ikat	27	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Kemiri
26	29	3	15000	Kg	29	https://tos.petra.ac.id/~c14170049/Skripsi/image/2...	Kunyit
26	30	10	1500	PCS	30	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Ketumbar
26	32	15	2000	PCS	32	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Pala
26	33	10	2000	PCS	33	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Jinten
26	31	10	5000	Ikat	31	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Wortel
26	35	10	3000	Ikat	35	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Bayam
26	36	2	60000	Kg	36	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Ikan Gurame
26	37	5	20000	PCS	37	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Ikan Tongkol
26	38	2	5000	PCS	38	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Ikan Pindang
26	39	5	10000	PCS	39	https://tos.petra.ac.id/~c14170049/Skripsi/image/3...	Cumi - Cumi
26	40	3	1500	PCS	40	https://tos.petra.ac.id/~c14170049/Skripsi/image/4...	Kangkung
26	41	5	3000	PCS	41	https://tos.petra.ac.id/~c14170049/Skripsi/image/4...	Selada Air
26	42	6	6000	PCS	42	https://tos.petra.ac.id/~c14170049/Skripsi/image/4...	Kubis

Gambar 5.22 Pengujian Tambah Barang Otomatis

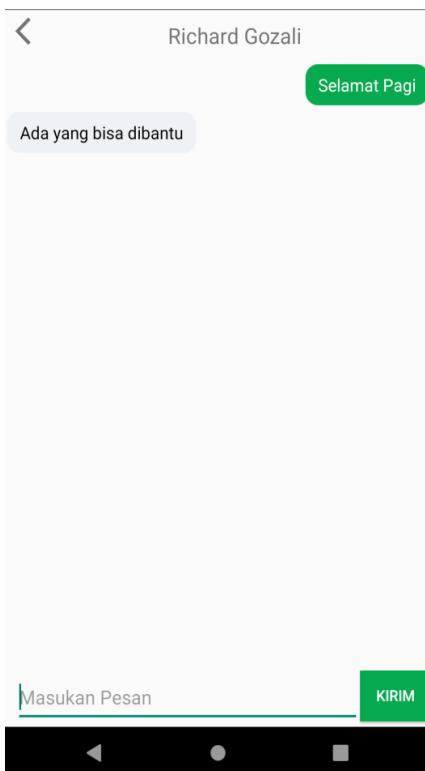
### 5.3.3 Pengujian Fitur Chatting

Proses pengujian terhadap fitur chatting pedagang keliling antar pelanggan. Berguna untuk berkomunikasi antar pedagang keliling dengan pelanggan. Setiap aktivitas yang terbuat akan ditampilkan tampilan *chatting*. Seperti pada Gambar 5.23 Tampilan *Detail Order* setelah tombol *chatting* dipencet. Akan keluar tampilan chat yang kemudian ditampilkan tersebut dapat dilakukanya komunikasi antara pedagang keliling dengan pelanggan.

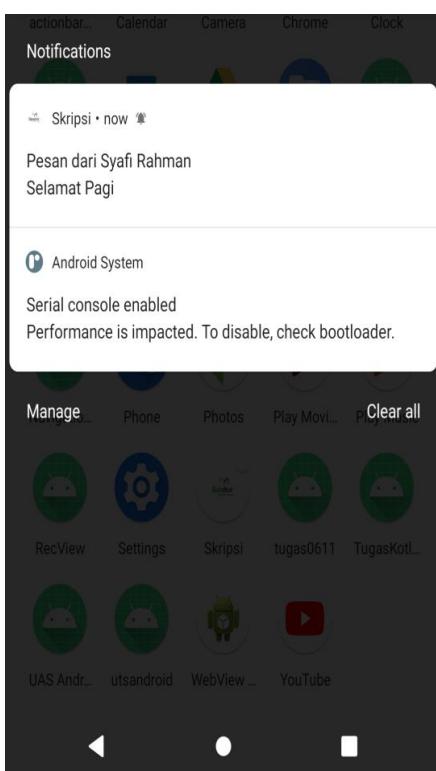
Setelah itu keduanya akan mendapatkan notifikasi jika salah satu diantaranya melakukan fitur *chatting*. Seperti pada Gambar 5.24 Hasil *chatting* dan *notifikasi* ini mampu membantu untuk mengetahui apakah *chatting* telah mendapat balasan dari pelanggan. Hasil dari fitur *chatting* tersebut pada pelanggan seperti pada Gambar 5.25 Hasil *Chatting* dan *Notifikasi Pelanggan*.



Gambar 5.23 Tampilan *Detail Order*



Gambar 5.24 Hasil *chatting* dan *notifikasi*



Gambar 5.25 Hasil *Chatting* dan *Notifikasi Pelanggan*

#### 5.4 Hasil Kuesioner Pelanggan

Untuk mengetahui penilaian pengguna pelanggan tentang program ini, dilakukan penelitian dan pengujian terhadap pelanggan. Dari hasil kuisioner yang telah dikumpulkan, didapatkan data seperti pada yang berisi tentang hasil kuisioner aplikasi tersebut. Pada Tabel 5.3 Hasil Kuesioner Pelanggan merupakan hasil dari Kuesioner pelanggan.

Tabel 5.3 Hasil Kuesioner Pelanggan

Nama Lengkap	Pertanyaan						
	1	2	3	4	5	6	7
Corry Margaretha	5	5	5	5	5	5	5
Arin Setyorini	5	4	5	4	4	5	5
Erny	4	5	5	5	4	5	5
Erna	4	4	4	5	5	4	5
Lina	5	4	5	5	4	5	4

Persentase penilaian untuk pertanyaan pertama tentang penilaian pelanggan terhadap tampilan aplikasi sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(2/5) *100\% = 40\%$
- nilai 5 =  $(3/5) *100\% = 60\%$

Persentase penilaian untuk pernyataan kedua tentang lokasi pedagang keliling yang dihasilkan cukup akurat sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(3/5) *100\% = 40\%$
- nilai 5 =  $(2/5) *100\% = 60\%$

Persentase penilaian untuk pertanyaan ketiga tentang fitur menu masakan berguna dan bermanfaat sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$

- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(2/5) *100\% = 40\%$
- nilai 5 =  $(3/5) *100\% = 80\%$

Persentase penilaian untuk pertanyaan keempat apakah fitur untuk mencari pedagang keliling mudah digunakan sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(1/5) *100\% = 20\%$
- nilai 5 =  $(4/5) *100\% = 80\%$

Persentase penilaian untuk pertanyaan kelima apakah fitur pembelian produk mudah untuk digunakan sebagai berikut

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(3/5) *100\% = 60\%$
- nilai 5 =  $(2/5) *100\% = 40\%$

Persentase penilaian untuk pertanyaan keenam apakah fitur *tracking* pedagang keliling berjalan dengan lancar sebagai berikut

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(1/5) *100\% = 20\%$
- nilai 5 =  $(4/5) *100\% = 80\%$

Persentase penilaian untuk pertanyaan ketujuh apakah aplikasi mudah untuk digunakan sebagai berikut

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(1/5) *100\% = 20\%$

- nilai 5 =  $(4/5) *100\% = 80\%$

Dari presentasi diatas ditarik kesimpulan bahwa :

- Tampilan aplikasi dinilai dengan 92%
- Lokasi yang ditampilkan untuk mengetahui pedagang keliling dinilai dengan 92%
- Fitur mencari menu makanan memiliki nilai 96%.
- Fitur untuk mencari pedagang keliling mudah untuk digunakan dengan nilai 96%
- Fitur pembelian produk dengan nilai 88%
- Fitur tracking pedagang keliling dengan nilai 96%
- Aplikasi untuk pelanggan mudah untuk digunakan dinilai dengan 96%

### 5.5 Hasil Kuesioner Pedagang Keliling

Untuk mengetahui penilaian pengguna pedagang keliling tentang program ini, dilakukan penelitian dan pengujian terhadap pedagang keliling. Dari hasil kuisioner yang telah dikumpulkan, didapatkan data seperti pada Tabel 5.4 Hasil Kuesioner Pedagang Keliling yang berisi tentang hasil kuisioner aplikasi tersebut

Tabel 5.4 Hasil Kuesioner Pedagang Keliling

Nama lengkap	Pertanyaan				
	1	2	3	4	5
Tri Juwono	3	5	4	5	4
Syafi Rahman	4	4	5	3	5
Slamet Febriansyah	4	5	5	4	5
Arjuna	5	4	5	4	5
Paulus Juwadi	5	5	4	5	5

Persentase penilaian untuk pertanyaan pertama tentang penilaian pedagang keliling terhadap tampilan aplikasi sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(1/5) *100\% = 20\%$
- nilai 4 =  $(2/5) *100\% = 40\%$

- nilai 5 =  $(2/5) *100\% = 40\%$

Persentase penilaian untuk pertanyaan kedua apakah data produk pedagang keliling telah mencukupi sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(2/5) *100\% = 40\%$
- nilai 5 =  $(3/5) *100\% = 60\%$

Persentase penilaian untuk pertanyaan ketiga apakah lokasi pelanggan akurat sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(2/5) *100\% = 40\%$
- nilai 5 =  $(3/5) *100\% = 60\%$

Persentase penilaian untuk pertanyaan keempat fitur *chatting* untuk pedagang keliling memudahkan untuk berkomunikasi dengan pelanggan sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(1/5) *100\% = 20\%$
- nilai 4 =  $(2/5) *100\% = 40\%$
- nilai 5 =  $(2/5) *100\% = 40\%$

Persentase penilaian untuk pertanyaan kelima apakah aplikasi untuk pedagang keliling mudah untuk digunakan sebagai berikut :

- nilai 1 =  $(0/5) *100\% = 0\%$
- nilai 2 =  $(0/5) *100\% = 0\%$
- nilai 3 =  $(0/5) *100\% = 0\%$
- nilai 4 =  $(1/5) *100\% = 20\%$
- nilai 5 =  $(4/5) *100\% = 80\%$

Dari presentasi diatas ditarik kesimpulan bahwa :

- Tampilan aplikasi dinilai dengan 84%
- Data produk keliling telah mencukupi dengan 92%

- Fitur lokasi pelanggan dinilai dengan 92%
- Fitur *chatting* memudahkan untuk berkomunikasi dinilai dengan 84%
- Aplikasi untuk pedagang keliling mudah untuk digunakan dinilai dengan 96%

## BAB 6 KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan yang diperoleh dalam Sistem Penunjang Belanja Pedagang Keliling di Lokasi Sekitar Menggunakan *Haversine* Berbasis *Android*. Selain itu, ada sejumlah saran untuk pengembangan lebih lanjut di masa yang akan datang.

### 6.1 Kesimpulan

Dari hasil pengujian sistem yang telah dilakukan, dapat diambil beberapa kesimpulan, diantar lain :

- Berdasarkan hasil pengujian kesimpulan yang diperoleh adalah penggunaan metode *scraping* untuk mengambil data menu masakan dari *eresep.com* berhasil dilakukan. Memakan waktu yang sebesar 4.776 detik.
- Penggunaan *Haversine* dalam penentuan jarak dianggap akurat. Dikarenakan telah dilakukan pengujian untuk mengetahui jarak yang dihasilkan oleh *google maps* dengan *haversine*, walaupun ketepatan penggunaan jarak *google maps* lebih akurat dengan menghitung jarak sesuai dengan jalan yang kita tempuh, kemacetan dan lain - lainya.
- Pengujian fitur untuk *filter* pedagang keliling berdasarkan lokasi berjalan dengan baik. Menampilkan semua pedagang keliling yang berjualan di sekitar area sebesar 1 km.
- Pengujian pembelian produk pedagang keliling telah berjalan dan data pun telah terkirim kedalam *database*. Pedagang Keliling telah mendapatkan data dari order pelanggan yang sesuai dengan keinginan pelanggan.
- Dari hasil pengujian sebanyak sepuluh kali dihasilkan rata-rata 3.775 detik untuk *upload* produk yang diperuntukan untuk disimpan kedalam *database*. Tergantung dengan kecepatan internet dan seberapa bagus *internal hardware* dari *smartphone* tersebut dan juga ukuran data gambar.
- Dari hasil pengujian fitur *chatting* telah dihasil alat komunikasi yang dapat berkomunikasi antar satu dengan yang lain. Dan juga dihasilkan *notifikasi* yang dihasilkan cukup baik untuk dapat mengetahui apakah pelanggan atau pedagang keliling telah membalas pesan yang telah dibuat.
- Berdasarkan hasil kuesioner dari pelanggan didapatkan jawaban dari pelanggan cukup baik. 40% setuju tampilan menarik dan 60% sangat setuju tampilan menarik. 40% setuju lokasi pedagang keliling akurat dan 60% sangat setuju lokasi akurat. 20% setuju dan 80% sangat setuju bahwa fitur mencari menu masakan bermanfaat. 40% setuju bahwa fitur pencarian pedagang keliling mudah dipakai dan 60% sangat setuju bahwa pencarian

pedagang keliling mudah untuk dipakai. 60% setuju bahwa fitur pembelian produk mudah untuk digunakan dan 40% sangat setuju bahwa pembelian produk mudah. 20% setuju dan 80% sangat setuju jika fitur tracking pedagang keliling berjalan dengan lancar. 20% setuju dan 80% sangat setuju bahwa aplikasi mudah untuk digunakan.

- Berdasarkan hasil kuisioner pedagang keliling didapatkan jawaban dari pedagang keliling cukup memuaskan. 20% pedagang keliling menilai tampilan dari aplikasi cukup. 40% dari pedagang keliling menilai tampilan aplikasi baik dan 40% pedagang keliling menilai sangat baik. Data produk yang telah disiapkan telah disiapkan dinilai baik. 40% menilai bahwa produk telah mencukupi dengan baik dan 60% menilai dengan sangat baik. Fitur untuk mengetahui posisi pelanggan dinilai cukup baik. 40% menilai bahwa lokasi yang dapat dinilai dengan baik dan 60 % dinilai dengan sangat baik. Fitur *chatting* untuk berkomunikasi dengan pelanggan dinilai 20% cukup. 40% dinilai baik dan 40% dinilai dengan sangat baik. Dari hasil kuesioner tersebut didapatkan bahwa 80% pedagang keliling menyetujui bahwa aplikasi yang digunakan mudah dengan sangat setuju dan 20% pedagang keliling menyetujui aplikasi mudah digunakan dengan setuju.

## 6.2 Saran

Saran yang dapat diberikan untuk melakukan penyempurnaan dan pengembangan program lebih lanjut, antara lain :

- Menambahkan fitur untuk mencatat apa saja bahan yang terdapat di dalam suatu menu masakan.
- Memperbaiki beberapa tampilan yang masih kurang sempurna pada aplikasi.
- Menambahkan situs penyedia menu masakan yang lain untuk *di-scraping*
- Data produk pedagang keliling yang dikumpul berasal dari lima pedagang keliling, Kemungkinan ada produk yang tidak terdata dan tidak termasuk.