Cyberspace: Jurnal Pendidikan Teknologi Informasi
Volume 5, Nomor 2, Oktober 2021, 158-172

APLLICATION OF HAVERSINE FORMULA IN NEAREST SALATIGA REPAIR SHOP SEARCH GEOGRAPHIC INFORMATION SYSTEM USING FLUTTER FRAMEWORK

Muhammad Arifin[1], Evangs Mailoa[2]

[1,2] Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
E-mail; marifin00@gmail.com , evangs.mailowa@uksw.edu

Abstract (11 pt, bold, at most 200 words)

The more vehicles that are used by Indonesian citizens, the risk of traffic accidents will also be greater. Many factors can affect the risk of traffic accidents, one of which is the abnormal condition of the vehicle used and a repair shop that can solve the problem is needed. By applying the Haversine formula and developing mobile application using Flutter, this research produces a geographic information system to find the closest distance from a motorbike and car repair shop in the city of Salatiga. This study produces a mobile application with a distance calculation that can be said to be close to accurate and displays the route from the user's point to the selected repair shop using the digital map builder from Google Maps.

Keywords: *Repair shop, Flutter, Dart, Haversine, Google Maps.*

Abstrak (11 pt, bold)

Semakin banyak nya kendaraan bermotor yang digunakan oleh warga Indonesia, resiko terjadi kecelakaan lalu lintas juga akan semakin besar. Banyak faktor yang dapat mempengaruhi resiko kecelakaan lalu lintas, salah satunya adalah tidak normalnya kondisi dari kendaraan yang digunakan dan diperlukan sebuah bengkel yang dapat mengatasi masalah tersebut. Dengan menerapkan formula *haversine* dan pengembangan aplikasi *mobile* menggunakan *Flutter,* penelitian ini menghasilkan suatu sistem informasi geografis pencarian jarak terdekat dari suatu bengkel motor dan mobil yang berada di kota Salatiga. Penelitian ini menghasilkan sebuah aplikasi *mobile* dengan perhitungan jarak yang dapat dikatakan mendekati akurat dan menampilkan rute dari titik pengguna ke bengkel yang dipilih dengan pembangun peta digital dari *Google Maps*.

Kata Kunci: *Bengkel, Flutter, Dart, Haversine, Google Maps.*

1. Pendahuluan

Indonesia merupakan salah satu negara dengan pertumbuhan kendaraan bermotor berjenis sepeda motor dan mobil yang pesat di dunia. Hal ini didukung data yang diperoleh dari Badan Pusat Statistik mengenai Jumlah Kendaraan Bermotor Menurut Jenis, yang menunjukkan pada tahun 2018 terdapat 120.101.047 unit kendaraan bermotor berjenis sepeda motor, dan 16.440.987 unit kendaraan bermotor berjenis mobil dan masih akan terus bertambah [1].

Tabel 1. Jumlah Kendaraan Bermotor dari Tahun 2015 s.d. 2018

Jumlah Kendaraan Bermotor (unit)				
Jenis Kendaraan	**2015**	**2016**	**2017**	**2018**
Motor	98.881.267	105.150.082	111.988.683	120.101.047
Mobil	13.480.973	14.580.666	15.423.968	16.440.987

Dengan berkembangnya kendaraan bermotor secara pesat, tentu bermacam resiko dalam berkendara juga tidak dapat dihindari, salah satunya adalah kecelekaan lalu lintas. Salah satu faktor dari sebuah kecelakaan lalu lintas adalah tidak normalnya kondisi suatu kendaraan dan dengan tujuan untuk mengembalikan performa kendaraan tersebut, maka perlu dilakukannya *tune up*. *Tune up* merupakan pengkondisian kembali atau pemulihan kerja motor sehingga kendaraan dapat kembali normal dengan pedoman yang ada dan dijadikan standar dalam *tune up* kendaraan bermotor [2]. Dengan minimnya informasi dan tata-cara *tune up* yang baik dan benar, maka pengguna kendaraan bermotor membutuhkan tempat yang dapat mengatasi masalah pada kendaraan mereka, salah satunya adalah bengkel. Selain penurunan performa, kendaraan bermotor juga sering didapati kebocoran ban di tengah perjalanan yang menyebabkan pengendara harus mendorong kendaraan nya atau bertanya kepada warga sekitar jika memungkinkan. Hal tersebut tentunya dapat menyulitkan para pengendara yang ingin segera sampai ke tempat tujuan secepatnya tetapi diharuskan untuk melakukan kegiatan yang membuang waktu dan tenaga[3]. Oleh karena dua permasalahan tersebut, pengguna kendaraan akan membutuhkan sesuatu yang bisa diandalkan disaat darurat untuk mencari bengkel motor atau mobil yang paling dekat, memanggil montir ke lokasi pengguna untuk melakukan *tune up* pada saat itu juga, atau memilih untuk membawa kendaraannya menuju ke lokasi bengkel yang paling dekat dari lokasinya.

Dalam penelitian ini, peneliti merancang suatu sistem informasi untuk mencari bengkel terdekat dari posisi pengendara yang menggunakan aplikasi. Data yang akan digunakan dalam penelitian ini adalah alamat dan titik koordinat dari beberapa bengkel yang ada di Kota Salatiga, dimana peneliti memperoleh lebih dari 50 data bengkel motor dan mobil yang ada di kota tersebut. Peneliti mengembangkan aplikasi *mobile android* menggunakan *framework Flutter* dengan bahasa pemrograman *dart* sebagai alat pencarian jarak bengkel terdekat. Tujuan menggunakan *Flutter* agar pembangunan aplikasi dilakukan lebih mudah, serta fitur fleksibilitas jika akan dikembangkan lagi ke sistem operasi *mobile iOS,* tidak perlu merubah seluruh *source code* dari program tetapi hanya menyesuaikan *library* nya saja.

Penggunaan algoritma atau formula *haversine* bertujuan agar dapat menghitung jarak antara titik di permukaan bumi menggunakan inputan variabel garis lintang (*longitude*) dan garis bujur (*latitude*) [4]. Peneliti memperoleh data berupa titik *latitude* dan *longitude* tersebut yang nantinya akan diolah menggunakan formual *haversine* dan mendapatkan hasil nilai jarak dengan satuan kilometer (km). *Output* yang dihasilkan oleh aplikasi ini berupa daftar nama, alamat, nomor telepon bengkel, serta diurutkan dari jarak yang terdekat. Lalu, pengguna dapat dengan mudah menemukan atau memilih bengkel yang paling cepat untuk dapat diakses.

2. Tinjauan Pustaka

Penelitian serupa dengan pencarian suatu penyedia layanan terdekat sudah dilakukan oleh beberapa peneliti. Penelitian yang berjudul Penerapan Formula Haversine Pada Sistem Informasi Geografis Pencarian Jarak Terdekat Lokasi Lapangan Futsal. Keterbatasan informasi tentang alamat, harga, dan kondisi lapangan yang dibutuhkan oleh pengguna masih sangat terbatas. Berdasarkan masalah yang sudah diajabarkan, penelitian tersebut merancang sistem yang dapat menampilkan informasi lapangan futsal dan dilengkapi dengan fitur pencarian jarak terdekat. Penelitian ini mennggunakan formula Haversine untuk menghitung jarak terdekat dari lapangan futsal dan menggunakan aplikasi Google Maps untuk membangun peta digital,lalu dikembangkan berbasis website [5].

Pada penelitian yang berjudul Aplikasi Pencarian Lokasi Spbu Terdekat Menggunakan Metode Algoritma Dijkstra Berbasis Android Di Kota Bandung, salah satu permasalahan terjadi ketika dalam perjalanan menggunakan kendaraan bermotor adalah habisnya bahan bakar. Karena hal tersebut pengguna kendaraan ingin mencari SPBU untuk mengisi kembali bahan bakar dari kendaraannya. Warga sekitar mungkin tidak akan menemukan masalah ketika diharuskan mencari lokasi SPBU terdekat. Tetapi akan terjadi masalah jika ada pengguna kendaraan bermotor yang bukan warga sekitar tetapi kehabisan bahan bakar. Penelitian ini menggunakan algoritma Djikstra dan mengembangkan suatu sistem berbasis aplikasi android untuk menyajikan data dari SPBU terdekat. Dari hasil pengujian, aplikasi ini sudah dapat dipakai untuk menentukan lokasi SPBU yang ada di kota Bandung [6].

Peneliti akan melakukan sebuah penelitian dengan mengembangkan sebuah aplikasi berbasis *mobile* yang dapat membantu pengendara sepeda motor atau mobil yang ingin melakukan perawatan ataupun memperbaiki masalah pada kendaraannya untuk menemukan bengkel yang paling dekat dengan lokasi pengguna yang ada di kota Salatiga. Aplikasi ini diharapkan dapat membantu para pengendara kendaraan bermotor di kota Salatiga untuk mencari bengkel terdekat dengan mudah dan ter-*install* di *smartphone* pengguna.

Penelitian aplikasi *mobile* ini dikembangkan menggunakan bahasa pemrograman *Dart* dengan *framework Flutter* untuk membantu pengguna kendaraan bermotor menemukan bengkel terdekat berdasarkan kendaraan yang dipakai. Lalu untuk penyajian data dari *database SQL* menggunakan *webservice* yang dikembangkan dengan bahasa pemrograman *PHP* yang nantinya akan dikirimkan ke dalam aplikasi *mobile* dalam bentuk *file JSON.*

Flutter

Flutter merupakan sebuah *framework open source* atau SDK dari *Google* untuk dapat mengembangkan aplikasi dengan sistem operasi *Android* dan *iOS*. Bahasa *dart* digunakan oleh *framework* ini dalam penulisan kode nya. *Flutter* memiliki perbedaan dengan bahasa pemrogramman lain yang dapat meng-*compile code* dalam *native-code* (seperti Android NDK, LLVM, AOT-compiled) dalam *build* aplikasi tanpa *interpreter* pada prosesnya, sehingga dapat lebih cepat dalam tahap *compile*-nya [7].

MYSQL

MySQL merupakan salah satu software RDBMS (database server) untuk memngelola dan membuat database secara otomatis dan terstruktur. Kemudahan yang diberikan oleh pengguna oleh MySQL adalah pada saat mengelola data yang berisi informasi String (text based), dapat diakses untuk umum maupun secara pribadi dalam suatu web. Akses MySQL dapat dilakukan oleh banyak pengguna (*multi-user*) dan diproses secara bersamaan serta diberikannya jumlah kapasitas yang besar dalam menampung data [8].

Webservice

Web Service merupakan logika aplikasi yang disusun untuk perangkat lain yang diolah menjadi layanan *web* agar data dari database dapat diakses melalui internet atau intranet. *Webservice* dapat memanipulasi data dalam mengambil, menambahkan atau mengubah data untuk melakukan pekerjaan tertentu sebagai sekumpulan dari berbagai fungsi. Dalam pertukaran data, *web service* biasa dibangun dalam format XML dan *Browser* atau *html* tidak diperlukan oleh *Web Service* karena tidak tersedianya *interface* untuk pengguna [9].

Web service mempunyai beberapa kelebihan sebagai berikut:

1. Proses pertukaran data lebih mudah. Data diantara dua aplikasi dapat dengan mudah dan cepat dalam pertukaran datanya, dikarenakan tidak harus menyesuaikan database dengan aplikasi yang digunakan.
2. Berbagai perangkat yang berbeda sistem operasi dapat saling bertukar data lintas lintas platform dengan *web service*.
3. Bahasa pemrograman apapun dapat mengakses sebuah *web service* dengan mudah (*Language Independent)*.
4. Jembatan antara aplikasi dan *database*. Aplikasi diberikan koneksi oleh untuk menghubungkan ke *database* agar data dapat dimanipulasi.

Haversine Formula

Haversine formula adalah persamaan yang penting pada navigasi, jarak lingkaran besar antara dua titik pada permukaan bumi diberikan berdasarkan bujur dan lintang. Efek *ellipsoidal* tidak diberi perhatian yang lebih atau diabaikan, yang menyebabkan cukup akuratnya perhitungan dari persamaan ini untuk sebagian besar perhitungan, lalu ketinggian bukit dan kedalaman lembah di sekitar diabaikan[5].

Persamaan (1) merupakan persamaan untuk menghitung *Haversine*

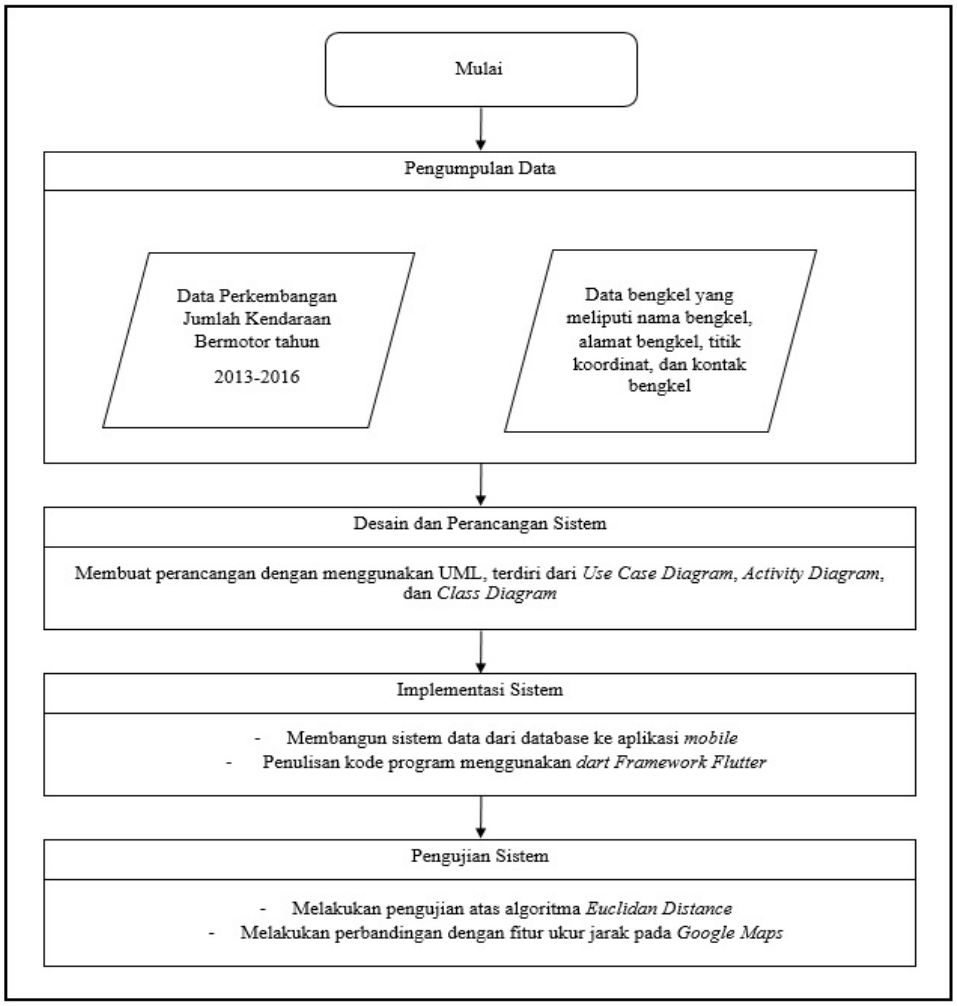
$\Delta lat = lat2 - lat1$
$\Delta long = long2 - long1$
$a = \sin2(\Delta lat/2) + \cos(lat1).\cos(lat2).\sin2(\Delta long/2)$
$c = 2\text{atan2}(\sqrt{\boldsymbol{a}}, \sqrt{\mathbf{1}-\boldsymbol{a}})$
$d = R.c$ (1)

Dimana :
R = jari-jari bumi sebesar 6371 (km)
Δlat = besaran perubahan *latitude*
Δlong = besaran perubahan *longitude*
c = kalkulasi perpotongan sumbu
d = jarak (km)
1 derajat = 0.0174532925 radian.

Nilai titik koordinat berbentuk *decimal degree* dan harus dikonversikan ke radian agar perhitungan menjadi sesuai dan tepat [10].

3. Metode Penelitian

Penelitian ini dibagi menjadi beberapa tahap, yaitu proses pengumpulan data, proses desain dan perancangan sistem, implementasi sistem, dan pengujian sistem.

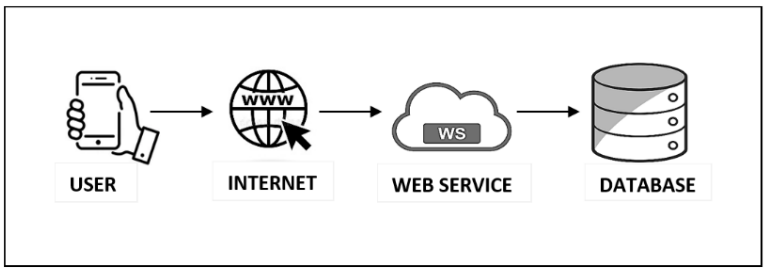


Gambar 1. Metode Penelitian

Gambar 1 dapat dijelaskan seperti berikut : Tahap pertama adalah identifikasi masalah yang bertujuan untuk mengetahui apa saja yang diperlukan oleh aplikasi dan nantinya dapat menunjang proses penelitian. Pada tahap pengumpulan data, peneliti mengumpulkan data yang akan menjadi acuan dalam proses perancangan aplikasi, yaitu data bengkel motor dan mobil yang ada di kota Salatiga dan diperoleh peneliti dengan rincian seperti nama bengkel, alamat bengkel, nomor telepon bengkel, titik lintang (*latitude)* dan titik bujur (*longitude*) dari bengkel yang nantinya nilai *latitude* dan *longitude* menjadi variabel utama perhitungan jarak bengkel terdekat.

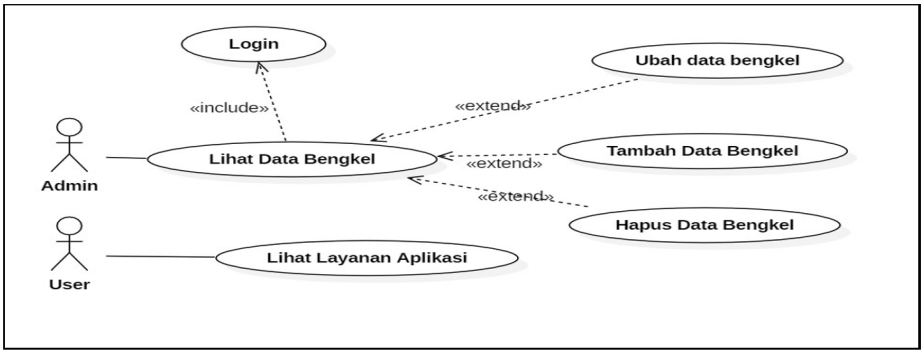
Pada tahap perancangan aplikasi, dilakukan pemodelan menggunakan UML untuk membuat alur program yang nantinya akan diimplementasikan menjadi program pada saat tahap penelitian implementasi sistem. Pemodelan yang digunakan oleh peneliti adalah *use case, activity,* dan *class diagram.* Setelah merancang aplikasi dengan UML, yang dilakukan peneliti sesudahnya adalah implementasi sistem, pada proses ini dilakukan pembuatan aplikasi pencarian bengkel terdekat dengan penulisan kode program sesuai dengan rancangan UML yang sudah dibuat, peneliti menggunakan kode pemrogramman *dart* dengan *framework Flutter* dari *google*. Setelah pembuatan kode program selesai, maka peneliti melanjutkan ke tahap pengujian sistem, dimana program akan dijalankan dan diuji apakah berjalan sesuai dengan rancangan aplikasi, berjalan dengan baik dan benar, untuk meminimalisir kesalahan dan mengetahui apakah output dari aplikasi sudah sesuai seperti yang diinginkan.

Dibawah ini adalah gambar serta penjelasan dari arsitektur sistem, dan beberapa diagram UML yang peneliti buat dan nantinya akan menjadi acuan dalam pengembangan aplikasi pencarian bengkel terdekat.



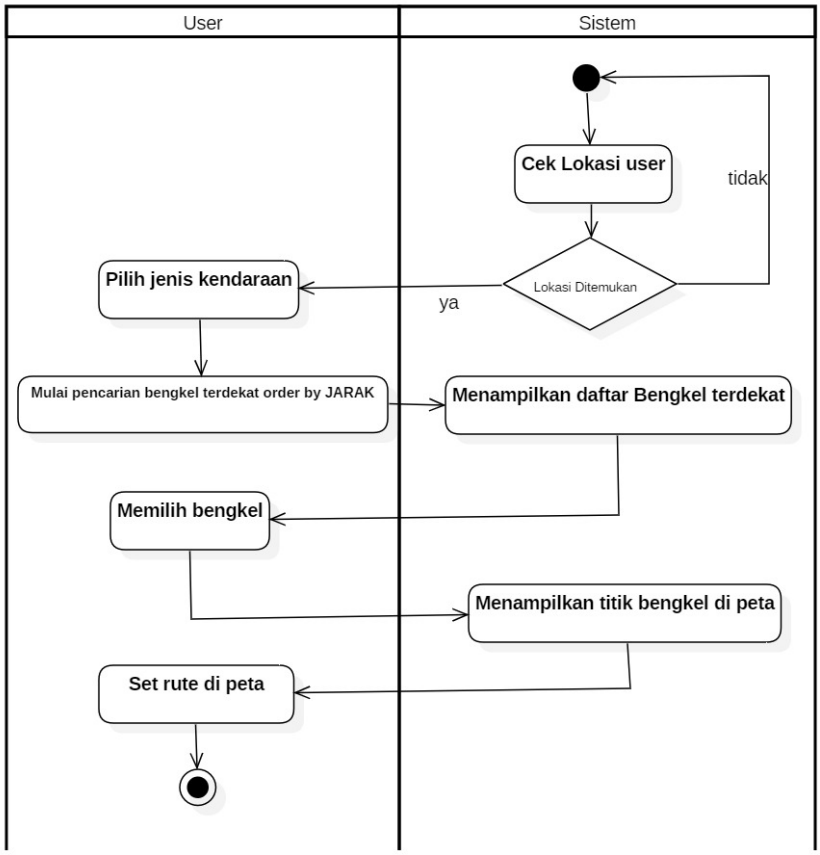
Gambar 2. Arsitektur sistem

Gambar 2 merupakan arsitektur sistem yang digunakan dalam aplikasi pencarian bengkel terdekat yang di implementasikan ke *smartphone* dari *user*. Untuk dapat berjalan, aplikasi memerlukan internet untuk mengkases *web service* yang disini berfungsi untuk menghubungkan antara aplikasi *mobile* dengan *database* yang berisi data bengkel yang terdaftar dalam bentuk file *JSON*.



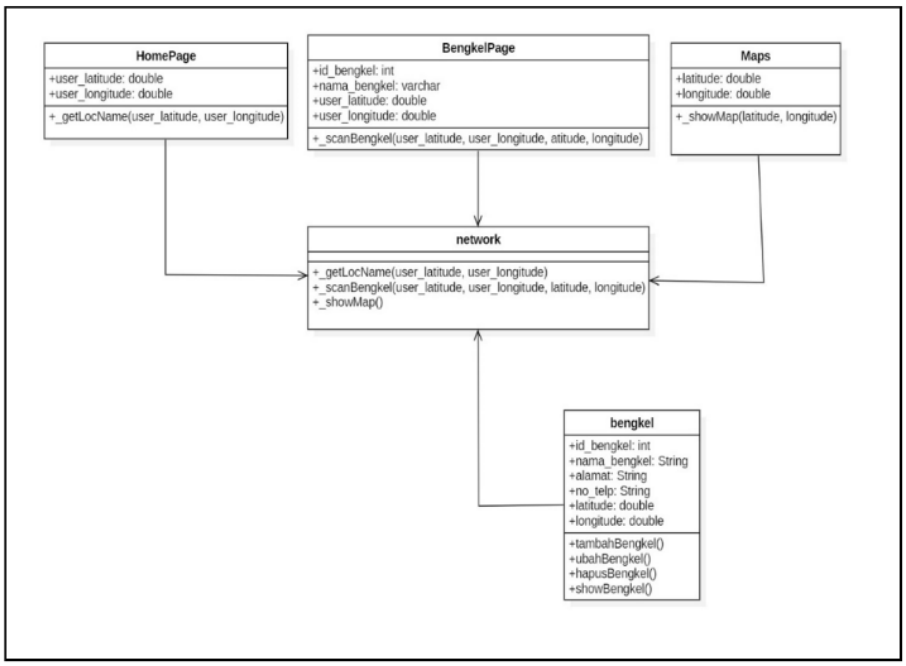
Gambar 3. *Use case diagram*

Gambar 3 adalah *Use case diagram* bertujuan menggambarkan fungsi yang bisa dijalankan di dalam sistem oleh entitas yang berperan di dalamnya, dalam hal ini yaitu *user* dan *admin.*



Gambar 4. *Activity diagram*

Gambar 4 merupakan *activity diagram* yang menunjukkan bahwa *user* dapat melakukan pencarian bengkel. Sistem akan terlebih dahulu mendeteksi lokasi titik lintang dan bujur (*latitude & longitude)* sebelum memuat tampilan dari aplikasi. Jika lokasi tidak diaktifkan pengguna maka kembali ke tahap awal dan memulai pencarian lokasi kembali. Lalu *user* bisa memulai mencari bengkel terdekat berdasarkan jenis kendaraan yang dipilih (motor/mobil), yang nantinya akan di urutkan oleh sistem dari jarak yang paling dekat hingga yang paling jauh. Setelah menentukan bengkel yang akan dituju, *user* bisa memilih dari daftar bengkel dan nantinya akan menuju aplikasi *GoogleMaps* untuk menunjukkan lokasi bengkel tersebut dan *user* bisa langsung mengetahui rute untuk kesana lewat fitur yang ada.



Gambar 5. *Class diagram*

Gambar 5 menjelaskan *class diagram* dari aplikasi pencarian bengkel terdekat. Bengkel yang sudah terdaftar di *database* mempunyai beberapa atribut seperti nama, alamat, nomor telepon, *latitude,* dan *longitude* yang nantinya akan digunakan sebagai bagian dari aplikasi. Lalu, *class network* berfungsi sebagai penampung fungsi-fungsi yang ada di dalam program dan nantinya akan digunakan dalam *class* HomePage, BengkelPage, dan Maps. Pada tahap implementasi sistem, peneliti melakukan penulisan dua kode program yaitu sebuah *web service* untuk menghubungkan data dari *database* dengan *client* dalam hal ini aplikasi *mobile* yang kode program nya ditulis dalam *framework Flutter* dan sesuai dengan rancangan peneliti yang sudah disusun. Lalu pada tahap terakhir yaitu pengujian sistem, dilakukan pengujian aplikasi yang dibangun lalu menganalisa data dari hasil penghitungan aplikasi dengan cara membandingkan dengan data jarak dari hasil perhitungan oleh *Google Maps.*

4. Hasil dan Pembahasan

Dalam penelitian ini, terdapat 55 (lima puluh lima) bengkel motor, dan 49 (empat puluh sembilan) bengkel mobil yang berada di kota Salatiga telah terdaftar pada *database*. Peneliti menghubungkan data dari *database* yang digunakan di dalam aplikasi *mobile* menggunakan *web service* dan data yang dihubungkan tersebut di konversi menjad *file JSON* untuk memudahkan peneliti dalam mengolah data tersebut.

Selanjutnya adalah langkah pencarian jarak dalam bentuk kilometer dari titik koordinat (*latitude* dan *longitude*) pengguna aplikasi yang didapat dari akses lokasi yang diberikan dan titik koordinat dari setiap bengkel motor atau mobil yang telah terdaftar menggunakan formula *haversine.* Jarak yang dihasilkan dari perhitungan tersebut lalu dibandingkan dan dan diurutkan dari nilai yang paling kecil yang berarti sudah didapatkan data jarak terdekat. Setelah ditampilkannya daftar bengkel motor atau mobil yang sudah dihitung jaraknya, pengguna bisa langsung meng-akses rute ke bengkel dengan aplikasi *Google Maps* yang telah dihubungkan ke sistem dengan memilih salah satu dari bengkel yang ditampilkan.

Kode Program 1 adalah kode program untuk meminta *user* untuk mengaktifkan lokasi *device* jika belum diaktifkan sebelumnya dan kemudian mendapatkan titik koordinat *user* untuk digunakan sebagai penghitungan jarak lokasi bengkel terdekat.

Kode Program 1. Kode program untuk mendapatkan titik koordinat *user*

```
1. location.requestPermission().then((granted) {
2.      if (granted != null) {
3.        location.onLocationChanged().listen((locationData) {
4.          if (locationData != null) {
5.            _locationController.add(UserLocation(
6.              latitude: locationData.latitude,
7.              longitude: locationData.longitude,));}});}});
```

Kode Program 2 merupakan kode program untuk model dari bengkel yang akan digunakan untuk menyimpan data *JSON* dari *database* melalui *web service.* Di bagian ini peneliti mengambil contoh dari model bengkel motor.

Kode Program 2. Kode program model bengkel

```
1.  int idBengkel;
2.  String namaBengkel;
3.  String alamat;
4.  String noTelp;
5.  double lat;
6.  double long;
7.  double jarak;

8.  factory BengkelMotor.fromJson(Map<String, dynamic> json) 9.   => BengkelMotor(
10.    idBengkel: json["id_bengkel"],
11.    namaBengkel: json["nama_bengkel"],
12.    alamat: json["alamat"],
13.    noTelp: json["no_telp"],
14.    lat: json["latitude"],
```

```
15.    long: json["longitude"],
16.    jarak: json["distance"]);
```

Untuk memasukkan data *JSON* ke dalam model bengkel motor diperlukan kode program seperti pada baris 8 sampai 16 dan disesuaikan dengan variabel yang akan diterima. Kode Program 3 merupakan kode program *web service* berbahasa pemrograman *PHP* untuk menghitung jarak bengkel terdekat dan mengirim data nama bengkel, alamat, nomor telepon dan hasil dari perhitungan jarak bengkel yang sudah dihitung.

Kode Program 3. Kode program *web service* untuk menghitung jarak.

```
1. <?php
2. header('Content-Type: application/json');
3. include dirname(dirname(__FILE__)).'/db/Db.class.php';
        $db = new Db();
4. $latitude = isset($_GET['lat']) ? (double) $_GET['lat']:0;
5. $longitude = isset($_GET['long'])?(double)$_GET['long']:0;
6. $databengkel = $db->
7. query('select*,(6371*acos(cos(radians('.$latitude.'))*
8. cos(radians(latitude))*cos(radians(longitude)-
9. radians('.$longitude.'))+
10. sin(radians('.$latitude.'))*sin(radians(latitude))))as
11. distance from motor order by distance asc');
12. echo json_encode($databengkel);
```

Web service tersebut memerlukan parameter yang dikirim dari *user,* yaitu titik koordinat dari *user* tersebut. Respon dari *web service* tersebut akan berbentuk *file JSON* dan berisi data bengkel yang sudah di urutkan dari jarak yang terdekat. Kode Program 4 adalah kode program untuk mendapatkan koneksi ke *web service* dari aplikasi yang hasilnya akan dikirimkan ke aplikasi dengan bentuk file *JSON*. Disini peneliti menggunakan contoh pengambilan data bengkel motor.

Kode Program 4. Koneksi aplikasi ke *web service*

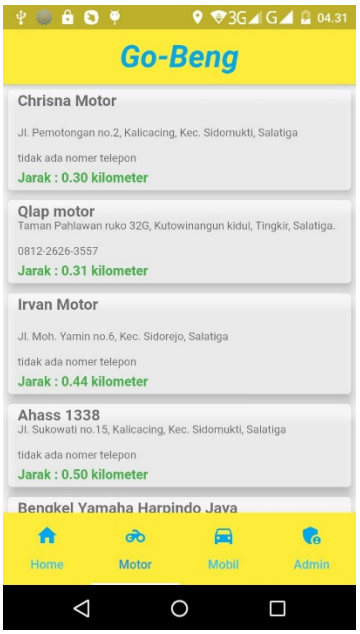
```
1. var hitungjarak =
2.'http://hitungjarakmotor.php/gobeng/api/categories/hitungja
3.rakmotor.php?lat=$lati&long=$longi';
4.    final response = await http.get(hitungjarak, headers: 5.headers);
6.    if(response.statusCode == 200){
7.      Iterable list = json.decode(response.body);
8.      var bMotor = list.map((bengkelMotor) =>
9.BengkelMotor.fromJson(bengkelMoto)).toList();
10.      return bMotor;    }
```

Pada baris 1 sampai 5 yang terdapat pada Kode Program 4, adalah kode program untuk mengirimkan parameter yang dibutuhkan *web service* untuk melakukan perhitungan jarak bengkel terdekat. Lalu, setelah file *JSON* berhasil didapatkan (mendapat kode respon 200), pada baris 6 sampai 9 berfungsi untuk memasukkan data ke dalam model yang sudah dibuat. Pada baris 10 terdapat fungsi *return* yang berfungsi mengembalikan data dalam bentuk *list* yang nanti digunakan sebagai variabel yang dapat ditampilkan pada halaman pencarian bengkel terdekat. Kode Program 5 merupakan kode program untuk menampilkan data yang sudah di simpan ke dalam model ke dalam halaman pencarian bengkel di aplikasi dalam bentuk daftar yang sudah diurutkan oleh *web service* dari jarak terdekat.

```
1.return Container(
2.      child: new FutureBuilder<List<BengkelMotor>>(
3.        future: _fetchBengkelMotor(userLocation.latitude,
4.userLocation.longitude),
5.        builder: (context, snapshot){
6.          if(snapshot.hasData){
7.            return RefreshIndicator(
8.              onRefresh: _onRefresh,
9.              color: Colors.blueAccent,
10.              child: ListView.builder(
11.                  itemCount: snapshot.data.length,
12.                  itemBuilder: (BuildContext context, int   13. index){
14.                    var sn = snapshot.data[index];
15.             var j=getDistanceFromLatLonInKm(sn.lat,sn.long);
16.                    BengkelMotor.setJarak(j);
17.                    return GestureDetector(
18.                      onTap: (){
19.                        MapsLauncher.launchCoordinates(
20.                            sn.lat, sn.long,
21. '${sn.namaBengkel}');
22.                      },
23.                      child: new Card(
24.                        elevation: 10,
25.                        color: Colors.white70,
26.                        margin: EdgeInsets.all(5),
27.                        child: new Column(
28.                          mainAxisAlignment:
29. MainAxisAlignment.center,
30.                          crossAxisAlignment:
31. CrossAxisAlignment.start,
32.                          children: <Widget>[
33.                            text(sn.namaBengkel, color: 34. Colors.black54,
35.                              size: 17.5,
36.                              isBold: true,
37.                              padding: EdgeInsets.only( 38. top: 2.0, left:
10.0),), text(sn.alamat, color:
39. Colors.black54,
40.                              size: 12.5,
41.                              padding: EdgeInsets.only( 42. left:
10.0),),text(sn.noTelp, color: Colors.black54,
43.size: 12.5,padding: EdgeInsets.only( left: 10.0),),
44.                      Column(
45.                              children: [
46.                                Row(children: [
47.                                  text('Jarak :
48. ${sn.jarak.toStringAsFixed(2)} kilometer', color: 49.Colors.green,size:
15,isBold: true,padding:
```

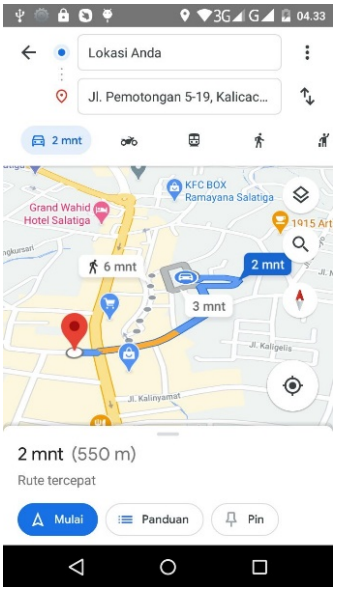
```
50. EdgeInsets.only(top: 5.0, left: 10.0),)],),
51. Padding(padding: EdgeInsets.only(bottom:
52. 5.0))],)],),),);}),);
53.           } else if (snapshot.hasError){
54.             print("count error :
55. ${snapshot.error.toString()}");
56.             return Column(children: [
57.  Center(child:text(snapshot.error.toString()),)],);
```

Baris ke tiga dari Kode Program 5 merupakan fungsi yang dipanggil untuk membangun sebuah *list* bengkel yang sudah diurutkan dari jarak yang terdekat. Fungsi tersebut mengirimkan parameter berupa *latitude* dan *longitude user* dan fungsi tersebut sudah ditampilkan dalam Kode Program 4. Pada baris 10 sampai baris 52 merupakan kode program yang digunakan untuk menempatkan masing-masing data yang akan didapat dari fungsi pada baris 3, yaitu nama bengkel, alamat bengkel, nomor telepon, dan jarak dalam satuan kilometer dari bengkel yang tertera ke titik lokasi *user.* Hasil dari perhitungan tersebut ditampilkan pada Gambar 6.



Gambar 6. Hasil pencarian bengkel terdekat

Pengujian aplikasi tersebut dilakukan di sekitar rumah peneliti dengan nilai *latitude* sebesar -7.327453, dan *longitude* 110.506048. Untuk memulai perjalanan menuju ke bengkel yang dipilih, *user* bisa men-*tap* pada salah satu pilihan bengkel tersebut dan akan langsung di arahkan ke aplikasi *Maps* yang ada di perangkat *user.* Tampilan peta setelah bengkel di pilih terdapat pada Gambar 7.



Gambar 7. Tampilan aplikasi peta setelah bengkel dipilih

Peneliti juga melakukan pengujian pencarian bengkel terdekat dari Universitas Kristen Satya Wacana untuk melakukan sampel analisa perhitungan. Berikut adalah contoh sampel perhitungan tersebut :

1. Titik koordinat pertama (Bengkel Mobil Montecarlo Salatiga)

Latitude 1 $= -7.30725 * \frac{\pi}{180}$
$= -0.12753478636$

Longitude 1 $= 110.489347 * \frac{\pi}{180}$
$= 1.92840289353$

2. Titik koordinat kedua (Universitas Kristen Satya Wacana)

Latitude 2 $= -7.317808 * \frac{\pi}{180}$
$= -0.12771984362$

Longitude 2 $= 110.499041 * \frac{\pi}{180}$
$= 1.92857208575$

3. $\Delta lat = -0.12771984362 - (-0.12753478636) = -0.00018505726$

4. $\Delta long = 1.92857208575 - 1.92840289353 = 0.00016919222$

5. $a = \sin2(\Delta lat/2) + \cos(lat1) * \cos(lat2) * \sin2(\Delta long/2)$
$= \sin2(-0.00018505726/2) + \cos(-0.12753478636) * \cos(-0.12771984362) * \sin2(0.00016919222)$
$= 0.000000267603482$

6. c = 2 * a sin($\sqrt{a}$)
= 2 * a sin($\sqrt{0.000000267603482}$)
= 0.000296393402
7. d = R * c
= 6371(km) * 0.000296393402
= 1.888322364 km

1,8 km adalah hasil dari contoh perhitungan koordinat bengkel mobil Montecarlo Salatiga ke koordinat *user* yaitu Universitas Kristen Satya Wacana. Perhitungan tersebut nantinya akan diterapkan pada masing-masing bengkel dan dipilih dari nilai terkecil sebagai lokasi bengkel terdekat dari titik koordinat *user.* Dengan perhitungan yang sama, peneliti melakukan pengujian lagi dengan membandingkan hasil pencarian bengkel motor dari aplikasi dengan hasil dari pencarian *Google Maps*. Perbandingan tersebut dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pencarian Jarak Bengkel Terdekat dari UKSW dan Perbandingan dengan Perhitungan *Google Maps*.

Nama Bengkel	Haversine Formula(km)	Google Maps (km)
Bengkel Ryan	0.2237870454252744	0.22332
Win Jaya Motor	0.32796306597828995	0.32951
Bengkel D-Ka	0.5182559393813561	0.51041
Hyper Speed	0.6019281665226244	0.60400
Tanjung Motor	0.6131013054427955	0.60813

Dari hasil perbandingan pada Tabel 2, perhitungan dengan formula *haversine* bisa dikatakan mendekati akurat jika dibandingkan dengan perhitungan fitur pengukuran jarak yang ada di *Google Maps* dan dapat dijadikan sebagai acuan *user* dalam pencarian bengkel berdasarkan jarak terdekat.

5. Penutup

Kesimpulan

Menurut hasil penelitian dan pengujian yang sudah dilakukan dapat disimpulkan bahwa aplikasi yang dikembangkan oleh peneliti sudah mendekati akurat jika dibandingkan dengan fitur pengukuran jarak yang terdapat di *Google Maps.* Sehingga pengguna dapat menggunakan aplikasi sebagai acuan untuk mencari bengkel yang paling terdekat dan tersedia dari titik lokasi dimana pengguna menggunakan *device*-nya. Aplikasi juga dapat membuka peta sesuai titik dari bengkel yang dipilih oleh pengguna.

Saran

Penelitian ini masih jauh dari kata sempurna sehingga diharapkan dapat dikembangkan lagi dengan menambah informasi tambahan seperti hari dan waktu dimana bengkel bisa didatangi. Dan mengingat formula *haversine* mengabaikan efek *ellipsoidal* yang tidak menghitung tinggi rendahnya suatu daratan, diharapkan juga aplikasi dapat dikembangkan lagi dengan algoritma atau formula yang lebih bisa menghitung efek tersebut.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik. (2016). Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis. https://www.bps.go.id/linkTableDinamis/view/id/1133. Diakses 11 November 2020.

[2] Ikhwanul Qiram, Ainoer Roffiq. (2017). Servis Sepeda Motor Gratis di Desan Pesucen Sebagai Upaya Peningkatan Skills Mahasiswa Teknik Mesin. Jati Emas (Jurnal Aplikasi Teknik dan Pengabdian Masyarakat) Vol.1 No. 1. hlm. 10

[3] Irawan, S., & Putra, G., H. (2016). Aplikasi Pencarian Tambal Ban Motor Terdekat Berbasis Android (Studi Kasus Tambal Ban di Kota Batam). Jurnal Teknomatika. Vol 9, no. 1, hlm 14.

[4] Prasetyo, D., & Hastuti, K. (2015). Penerapan Haversine Formula Pada Aplikasi Pencarian Lokasi Dan Informasi Gereja Kristen Di Semarang Berbasis Mobile. Skripsi Teknik Informatika Universitas Dian Nuswantoro.

[5] Yulianto. Ramadiani. Awang Harsa Kridalaksana. (2018). PENERAPAN FORMULA HAVERSINE PADA SISTEM INFORMASI GEOGRAFIS PENCARIAN JARAK TERDEKAT LOKASI LAPANGAN FUTSAL. Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer. Vol. 13, No. 1 Februari 2018. hlm 18.

[6] Nawam Milal, Sri Nurhayati. Aplikasi Pencarian Lokasi Spbu Terdekat Menggunakan Metode Algoritma Dijstrak Berbasis Android Di Kota Bandung. Skripsi Teknik Komputer Universitas Komputer Indonesia.

[7] Dian, Muhar. (2018). Tutorial Flutter #1: Pengenalan dan Persiapan Pemrograman Mobile dengan Flutter. [Online]. Tersedia: https://www.petanikode.com/flutter-linux/.

[8] Widad, Muhamad Faisal. Sistem Informasi Penjualan Berbasis Android Di Toko Busana Faisal Collection (SIPITON). Program Studi Sistem Informasi. Fakultas Teknik dan Ilmu Komputer. Universitas Komputer Indonesia. Bandung. 2017.

[9] Anindita. (2015). Pengenalan Web Service. [Online]. Tersedia: https://aninditasaktiaji.com/pengenalan-web-service/.

[10] Rubin A. Geo (proximity) Search with MySQL 2006. http://www.arubin.org/files/geo_search.pdf. Diakses tanggal 17 November 2020.