Vol. 5 No. 1, 2024, pp. 83-92 **Jurnal Pepadun**

Haversine Geo-Spasial Data Android Model Untuk Optimasi Rute Kebersihan Lingkungan Terdekat

[1,2*] Riko Herwanto, [3]Ferry Susanto, [4]Rosmala Dwi, [5]M. Husain Prayoga, [6]Riadi Marta Dinata dan [7]Wamiliana

[1,3,5,6]Program Doktoral FMIPA, Universitas Lampung, Jl. Prof. Dr. Sumantri Brojonegoro, Bandar Lampung, Indonesia
[2]Teknik Informatika, Institut Informatika dan Bisnis Darmajaya, Jl. ZA Pagaralam No. 93, Bandar Lampung, Indonesia
[4]Teknik Informatika, Universitas Nahdlatul Ulama Lampung, Taman Fajar, Purbolinggo, Lampung Timur, Indonesia
[8]Prodi Matematika, FMIPA, Universitas Lampung, Jl. Prof. Dr. Sumantri Brojonegoro, Bandar Lampung, Indonesia

e-mail: ***[1]2337061003@student.unila.ac.id**, ***[2]rikoherwanto@darmajaya.ac.id**, [3]ferrysusanto80@gmail.com, [4]rosmaladwi@unulampung.ac.id, [5]m.hprayoga9@gmail.com, [6]riadi.mrt@gmail.com, [7]wamiliana.1963@fmipa.unila.ac.id

Abstract *— The urgency of environmental sustainability necessitates innovative solutions to address growing waste management challenges. This research investigates the development of a mobile application that utilizes geospatial data and the Haversine formula to optimize waste disposal efficiency in Indonesia. The dramatic increase in national waste generation, from 65.8 million tons in 2017 to 75.2 million tons in 2018, highlights the critical need for such interventions. The application leverages geotagging to pinpoint the nearest and most suitable waste disposal locations, streamlining the waste collection process for environmental workers. To ensure accurate distance calculations, the Haversine formula is employed, achieving an 84% accuracy rate with an average deviation of 112.522 meters. Black-box testing methodologies evaluate the application's interface functionality, while data-source optimization is explored to enhance application speed and waste collection route planning. This research contributes to the growing body of knowledge concerning technology-driven solutions for sustainable waste management practices.*

Keywords**:** *Geo-Tagging; Haversine; Mobile; Sort-Route.*

1. PENDAHULUAN

1.1. Permasalahan Secara Umum

Lingkungan hidup kini menjadi isu global yang mendapat perhatian luas. Kondisi lingkungan yang semakin memburuk dipicu oleh pertumbuhan populasi yang pesat, keterbatasan sumber daya alam, dan eksploitasi alam yang tidak bertanggung jawab. Tindakan-tindakan yang ceroboh terhadap lingkungan menjadi penyebab utama berbagai permasalahan lingkungan baik di tingkat global maupun nasional. Temuan ini didasarkan pada penelitian dan observasi yang dilakukan selama bertahun-tahun. Mayoritas kerusakan lingkungan terjadi akibat aktivitas manusia yang memposisikan diri mereka terpisah dari alam dan menjadikan alam sekadar objek dalam sistem ekologis. [1].

Peningkatan konsumsi masyarakat berbanding lurus dengan pertumbuhan ekonomi yang pesat. Masyarakat kini tidak hanya memanfaatkan sumber daya alam untuk memenuhi kebutuhan dasar mereka, tetapi juga melampaui kebutuhan tersebut. Hal ini mengakibatkan banyaknya barang-barang tak terpakai yang berpotensi merusak lingkungan. Sampah yang dibuang sembarangan menjadi masalah serius, karena dapat menyebabkan banjir, pencemaran lingkungan, polusi udara, dan berbagai masalah lainnya [2]. Berdasarkan observasi dan diskusi dengan petugas kebersihan, masalah utama dalam menjaga kebersihan lingkungan adalah kurangnya pemahaman masyarakat tentang pentingnya upaya pembersihan. Penempatan tempat sampah disesuaikan dengan jalur petugas kebersihan yang terdekat [3]. Namun, petugas kebersihan sering kali tidak memiliki informasi tentang lokasi tempat sampah yang sesuai dengan rute mereka. Untuk mengatasi masalah ini, metode pencarian rute terdekat dapat diimplementasikan dengan menggunakan algoritma Haversine dan teknologi *geo-tagging* untuk membantu petugas kebersihan menemukan jalur optimal dalam pembersihan lingkungan [4].

Navigasi dan eksplorasi lingkungan sekitar semakin penting di era digital ini. Masyarakat mencari cara yang lebih efisien dan informatif untuk menemukan tempat-tempat menarik, rute terbaik, dan informasi terkait lingkungan di sekitar. Aplikasi peta dan navigasi tradisional seringkali tidak memberikan informasi yang cukup detail dan kontekstual, terutama terkait dengan lingkungan dan amenitas lokal.

1.2. Masalah Secara Khusus

Model Haversine Android yang umum digunakan untuk perhitungan jarak terdekat dalam aplikasi navigasi memiliki beberapa keterbatasan:

a. Kurangnya informasi kontekstual: Model ini hanya mempertimbangkan jarak geografis, tanpa memperhitungkan faktor-faktor lain seperti kondisi jalan, medan, aksesibilitas, dan amenitas di sepanjang rute.
b. Ketidaktepatan dalam beberapa situasi: Model haversine berasumsi bahwa bumi berbentuk bola sempurna, yang dapat menyebabkan ketidaktepatan dalam perhitungan jarak, terutama di daerah dengan medan yang kompleks.
c. Kurangnya personalisasi: Model ini tidak mempertimbangkan preferensi pengguna dan kebutuhan spesifik mereka, seperti preferensi moda transportasi, tingkat kebugaran, atau minat pribadi.

1.3. Solusi yang Sudah Ada

Beberapa solusi telah dikembangkan untuk mengatasi keterbatasan model Haversine Android, yaitu:

a. Penggunaan data peta yang lebih detail. Integrasi data peta yang lebih detail, seperti informasi jalan, medan, dan amenitas, dapat meningkatkan akurasi perhitungan jarak dan memberikan informasi kontekstual yang lebih kaya.
b. Penerapan algoritma perutean yang lebih canggih. Algoritma perutean yang mempertimbangkan faktor-faktor seperti kondisi lalu lintas, waktu tempuh, dan preferensi pengguna dapat memberikan rute yang lebih optimal dan efisien.
c. Personalisasi berdasarkan profil pengguna. Pertimbangan profil pengguna, seperti preferensi moda transportasi, tingkat kebugaran, dan minat pribadi, dapat menghasilkan rute yang lebih sesuai dengan kebutuhan dan keinginan individu.

1.4. Kekurangan/Kelebihan dari Solusi yang Sudah Ada

Algoritma Haversine digunakan dalam aplikasi yang memberikan informasi tentang navigasi jarak terpendek. *Geotagging* menambahkan informasi posisi data ke *Global Positioning System* (GPS) berupa informasi lintang dan bujur pada suatu gambar. Solusi yang ada memiliki beberapa kelebihan dan kekurangan. Kelebihannya mencakup 3 hal, yaitu: (1) meningkatkan akurasi perhitungan jarak dan informasi kontekstual; (2) memberikan rute yang lebih optimal dan efisien; dan (3) menawarkan pengalaman navigasi yang lebih personal. Di sisi lain terdapat kekurangan yakni: (1) memerlukan data peta yang lebih detail dan infrastruktur komputasi yang lebih kuat; (2) algoritma perutean yang kompleks dapat meningkatkan waktu pemrosesan; serta (3) personalisasi yang berlebihan dapat membatasi pilihan pengguna.

1.5. Solusi yang Akan Diberikan

Penelitian ini bertujuan untuk mengembangkan model *geotagging* Android haversine yang disempurnakan untuk navigasi lingkungan sekitar. Model ini akan menggabungkan beberapa solusi yang ada, dengan fokus pada tiga aspek sebagai berikut:

a. Integrasi data peta yang komprehensif. Model ini akan mengintegrasikan data peta yang detail dan terkini, termasuk informasi jalan, medan, amenitas, dan kondisi lalu lintas.
b. Penerapan algoritma perutean yang canggih. Algoritma perutean yang mempertimbangkan faktor-faktor seperti waktu tempuh, kondisi jalan, dan preferensi pengguna akan digunakan untuk menghasilkan rute yang optimal.

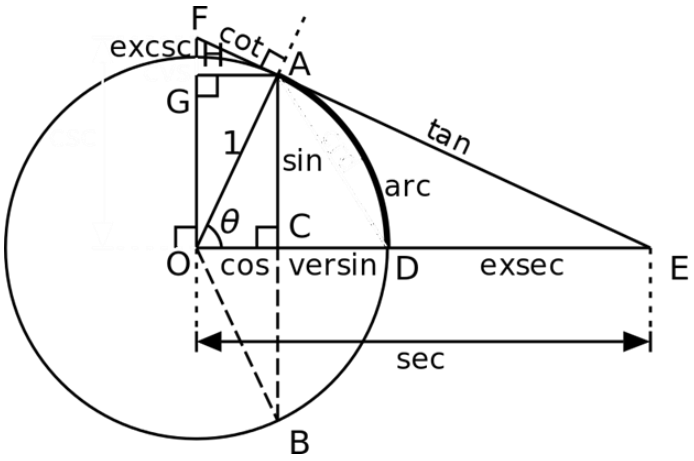
c. Personalisasi berdasarkan profil dan konteks pengguna. Model ini akan mempertimbangkan profil pengguna, seperti preferensi moda transportasi, tingkat kebugaran, dan minat pribadi, serta konteks saat ini, seperti waktu dan cuaca, untuk memberikan rekomendasi rute yang tepat dan relevan.

Model *geo-tagging* Android haversine yang disempurnakan ini diharapkan dapat memberikan solusi navigasi yang lebih akurat, informatif, dan personal bagi pengguna, meningkatkan pengalaman eksplorasi dan navigasi lingkungan sekitar mereka.

2. METODOLOGI PENELITIAN

2.1 Algoritma Haversine

Haversine merupakan persamaan integral dalam sistem navigasi [5]. Rumus Haversine menghitung jarak terpendek antara dua titik, misalnya bujur dan lintang, pada sebuah bola [6], dimana merupakan penerapan konsep trigonometri yang merupakan bagian dari geometri. Rumus Haversine adalah persamaan navigasi penting yang menunjukkan jarak lingkaran besar antara dua titik (lintang dan bujur) di permukaan bola (bumi) sebagai fungsi dari garis bujur dan garis lintang. Penggunaan metode ini dipakai tanpa memperhatikan ketinggian permukaan bukit dan kedalaman lembah [7][8].



Gambar 1. Pola Algoritma Haversine [5].

Gambar 1 mengilustrasikan pola rumus Haversine yang dijelaskan dalam konteks trigonometri bola. Persamaan ini sangat penting dalam bidang navigasi, karena mampu menentukan jarak terpendek antara dua titik. Awalnya, rumus Haversine digunakan untuk mengatasi masalah utama dalam astronomi bahari, khususnya untuk mengukur jarak antar bintang [8]. Josef de Mendoza y Rios pertama kali memanfaatkan rumus ini pada tahun 1801, sedangkan James Andrew menemukan rumus tersebut pada tahun 1805. Istilah "haversine" sendiri diperkenalkan oleh Prof. James Inman pada tahun 1835. Dengan asumsi bahwa bumi berbentuk bulat sempurna dengan radius 6.3671 km dan kedua titik berada pada koordinat bola (lintang dan bujur) yang dinyatakan sebagai lon1, lat1 dan lon2, lat2, maka persamaan Haversine dapat ditulis seperti pada Persamaan 1. Persamaan Haversine merupakan persamaan pertama yang perlu diperhatikan saat menghitung jarak pada sebuah bola [9]. Kata "Haversine" berasal dari bentuk fungsi yang terdapat pada persamaan.

$$\boldsymbol{haversine}\ (\boldsymbol{\theta}) = \boldsymbol{sin^2}\left(\frac{\boldsymbol{\theta}}{\boldsymbol{2}}\right) \quad (1)$$

Notasi φ adalah garis lintang, λ adalah garis bujur, dan R adalah jari-jari bumi (rata-rata radius = 6,371km) adalah seluruh notasi yang merepresentasikan bagaimana menerjemahkan persamaan tersebut untuk memasukkan koordinat garis lintang dan garis bujur. Perhatikan bahwa sudut harus dalam satuan radian agar dapat digunakan ke fungsi trigonometri sebagaimana pada Persamaan 2 dan 3.

$$\boldsymbol{a = sin^2 + \left(\varphi B - \frac{\varphi A}{2}\right) + \cos \varphi A \times \cos \varphi B \times \left(\lambda B - \frac{\lambda A}{2}\right)} \quad (2)$$

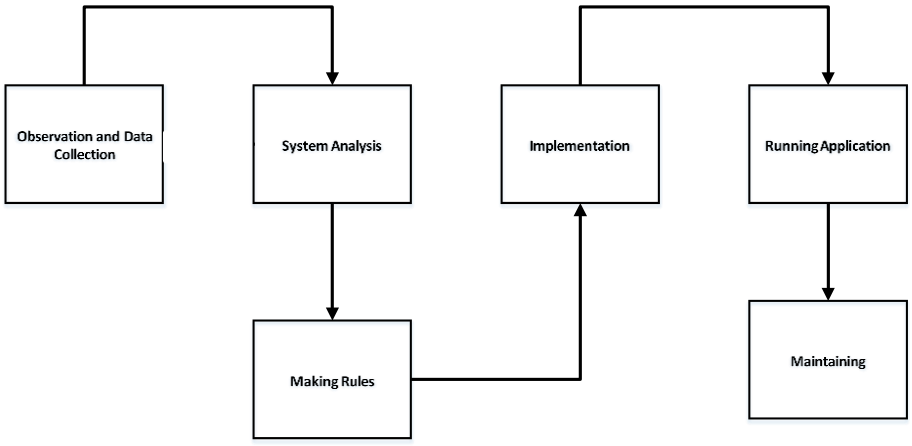
$$\boldsymbol{c = 2 \times atan\ 2 \left(\sqrt{a}, \sqrt{(1-a)}\right)} \quad (3)$$

$$\boldsymbol{d = R \times c} \quad (4)$$

2.2 *Geo-Tagging*

Penelitian [10] menyatakan bahwa "*geo-tagging*" atau yang juga disebut sebagai "*geo-referencing*" adalah proses menambahkan metadata ke gambar dan video yang memberitahukan lokasi pengambilannya. Proses ini dapat membantu pengguna menemukan berbagai macam informasi mengenai suatu lokasi. *Geo-tagging* manual adalah metode dimana informasi tentang lokasi ditambahkan secara manual dengan memasukkan koordinat tertentu atau memilih lokasi saat mengunggah media ke internet [11]. Tingkat akurasi metode *geo-tagging* ini bergantung pada alat yang digunakan atau penerima GPS untuk mendapatkan koordinat yang akurat [12].

Google Map API merupakan suatu alat atau penyedia layanan yang disediakan oleh teknologi bernama Google kepada penggunanya agar dapat memanfaatkan Google Maps ketika mengembangkan aplikasi yang dibangun [13]. Google Maps API juga memiliki cara berbeda untuk mendapatkan data dan menambahkan kontak melalui penyedia layanan berbeda, dan memungkinkan lebih dari satu pengguna mengakses data. Desain penelitian merupakan suatu konsep atau gambaran penelitian yang akan dilakukan. Gambaran alur penelitian dapat dilihat pada Gambar 2.



Gambar 2. Rancangan Penelitian.

2.2.1. Observasi dan Pengumpulan Data.

Observasi dan pengumpulan data merupakan fungsi pengumpulan data dimana dilakukan pencatatan segala informasi yang dikumpulkan selama penelitian berlangsung, yaitu suatu metode pengumpulan data dengan cara pengamatan langsung terhadap situasi atau kejadian di lapangan.

2.2.2. Analisis Sistem

Setelah berkonsultasi dengan pengguna, maka layanan, batasan, dan tujuan sistem ditentukan pada tahap ini. Semuanya didefinisikan secara rinci dan dijadikan spesifikasi sistem dari observasi dan pengumpulan data yang dilakukan pada penelitian.

2.2.3. Membuat Aturan.

Proses pembuatan aturan menyediakan kebutuhan perangkat keras atau perangkat lunak dengan menyediakan arsitektur sistem secara keseluruhan. Desain sistem melibatkan identifikasi dan deskripsi abstraksi sistem.

2.2.4. Penerapan

Pada tahap ini perancangan sistem diwujudkan sebagai suatu program atau unit program. Pengujian unit melibatkan verifikasi apakah setiap unit memenuhi spesifikasi sistem.

2.2.5. Menjalankan Aplikasi

Setelah unit program dirancang, unit program tersebut akan diuji untuk menjalankan aplikasi yang melibatkan verifikasi untuk memastikan setiap unit memenuhi spesifikasi sistem.

2.2.6. *Maintaining*

Pemeliharaan menginstal sistem dan menggunakannya dalam praktik. Pemeliharaan berarti memperbaiki kesalahan yang tidak ditemukan pada tahap sebelumnya, meningkatkan implementasi unit sistem, dan meningkatkan layanan yang ditawarkan sistem ketika ditemukan kebutuhan baru.

Penggabungan algoritma Haversine dengan *geo-tagging* melibatkan beberapa langkah sebagai berikut.

a. Mengumpulkan data geografis. Langkah pertama adalah mengumpulkan data geografis, seperti koordinat lintang dan bujur, untuk setiap lokasi yang ingin disertakan dalam aplikasi atau layanan. Data ini dapat diperoleh dari perangkat GPS, peta, atau sumber lainnya [10].
b. Implementasi algoritma Haversine. Langkah selanjutnya adalah mengimplementasikan Algoritma Haversine pada aplikasi atau layanan. Algoritma Haversine menggunakan koordinat lintang dan bujur untuk menghitung jarak antara dua titik pada permukaan bola. Kode algoritma Haversine sesuai dengan bahasa pemrograman yang dipakai [11]. Berikut adalah implementasi Python sederhana dari algoritma Haversine dengan *geo-tagging*.

```
import math
def haversine(lat1, lon1, lat2, lon2):
"""
Calculates the distance between two points on the earth using the Haversine
formula.
"""
R = 6371 # Earth's radius (in km) dLat = math.radians(lat2 - lat1) dLon =
math.radians(lon2 - lon1)
a = math.sin(dLat/2) * math.sin(dLat/2) + math.cos(math.radians(lat1)) *
math.cos(math.radians(lat2)) * math.sin(dLon/2) * math.sin(dLon/2)
c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a)) d = R * c
return d
def get_distance(point1, point2):
"""
Calculates the distance between two points on the earth using the Haversine
formula.
"""
lat1, lon1 = point1['latitude'], point1['longitude'] lat2, lon2 =
point2['latitude'], point2['longitude'] return haversine(lat1, lon1, lat2, lon2)
 # Example usage
point1 = {'latitude': 37.7749, 'longitude': -122.4194}
point2 = {'latitude': 51.5074, 'longitude': -0.1278}
distance = get_distance(point1, point2) print(f"Distance between points:
{distance} km")
```

c. Menyimpan data geografis. Setelah mengumpulkan data geografis dan menerapkan algoritma Haversine, langkah selanjutnya adalah menyimpan data tersebut dalam *database* atau sistem penyimpanan data lainnya. Penyimpanan dapat menggunakan *database* NoSQL, seperti MongoDB atau Cassandra, atau *database* relasional, seperti MySQL atau PostgreSQL.
d. Menambahkan *geo-tagging*. Langkah terakhir adalah menambahkan *geo-tagging* pada aplikasi. Koordinat lintang dan bujur digunakan untuk menambahkan metadata identifikasi geografis ke berbagai media seperti foto, video, dan situs web.

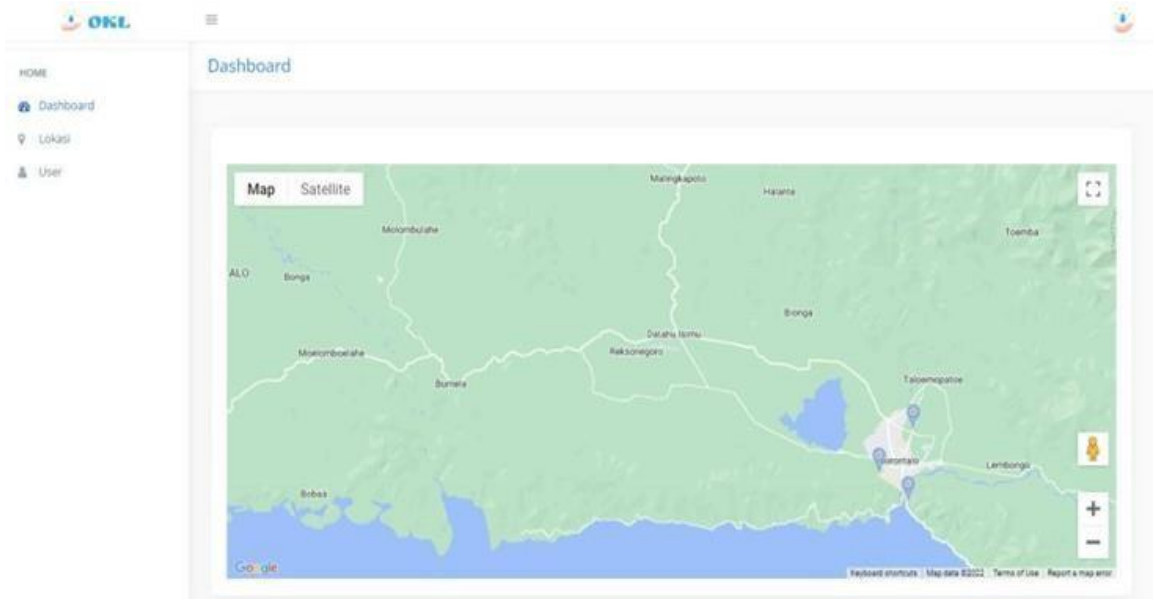
Dengan mengikuti langkah-langkah tersebut, maka digabungkan algoritma Haversine dengan *geo-tagging* untuk membuat layanan dan aplikasi berbasis lokasi yang dapat menentukan jarak antara dua titik secara akurat dan mengaitkan informasi tersebut dengan media tertentu atau data lainnya. Untuk menghitung keakuratan

penggabungan algoritma Haversine dengan *geo-tagging* [12], dilakuan perbandingan hasil yang diperoleh dari kombinasi tersebut dengan data *ground truth* sebenarnya. Berikut adalah garis besar umum proses yang dilakukan.

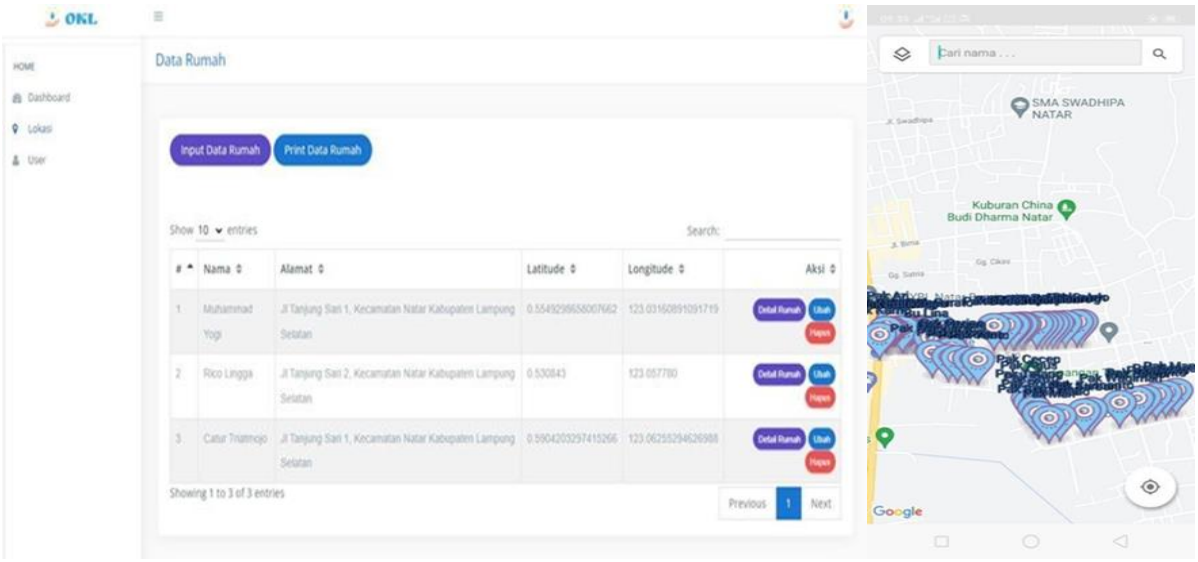
a. Memperoleh data kebenaran dasar. Proses ini melibatkan perolehan data dunia nyata yang akurat untuk lokasi yang ingin dibandingkan dengan hasil. Perolehan data dilakukan melalui berbagai cara, seperti survei lapangan, pengukuran GPS, atau dengan menggunakan lokasi referensi yang diketahui dan akurat.
b. Penggunaan algoritma Haversine dengan *geo-tagging*. Pada tahap ini dihitung jarak antar lokasi menggunakan algoritma Haversine (*Placeholder2*) dengan koordinat *geo-tagged*.
c. Perbandingan hasil dengan data kebenaran dasar. Pada proses ini dibandingkan hasil yang diperoleh dari algoritma Haversine dengan data kebenaran dasar. Hal ini dapat dilakukan dengan menghitung selisih antara dua kumpulan data.
d. Evaluasi hasil dengan akurasi. Evaluasi dilakukan dengan menghitung akurasi menggunakan metrik seperti *Root Mean Squared Error* (RMSE), yang mengukur deviasi rata-rata hasil dari data kebenaran dasar. Nilai RMSE yang rendah menunjukkan akurasi yang tinggi, sedangkan nilai RMSE yang tinggi menunjukkan akurasi yang rendah.
e. Pengulangan proses. Proses tersebut diulang beberapa kali dengan kumpulan data yang berbeda untuk mendapatkan nilai akurasi rata-rata.
f. Setelah melalui rangkaian proses berikut, dihitung keakuratan penggabungan algoritma Haversine dengan *geo-tagging*. Langkah spesifik dan detailnya mungkin berbeda-beda, bergantung pada data yang dimiliki dan kasus penggunaan spesifik, namun hal ini memberikan gambaran umum tentang bagaimana menghitung keakuratan penggabungan kedua teknologi ini [13].

3. HASIL DAN PEMBAHASAN

Pengujian jarak diperlukan dalam penelitian ini untuk mengetahui keakuratan penghitungan jarak dengan menggunakan metode Haversine. Pengujian dilakukan dengan membandingkan perhitungan jarak antara perhitungan Haversine dengan perhitungan yang diberikan oleh aplikasi Google Maps. Rangkaian proses pengujian tersebut direpresentasikan melalui Gambar 3, 4 dan 5.



Gambar 3. Hasil Implementasi API *Database*.



Gambar 4. Hasil Implementasi *Database Geo-tagging* dan Algoritma Haversine.

3.1. Hasil Pengujian Antarmuka

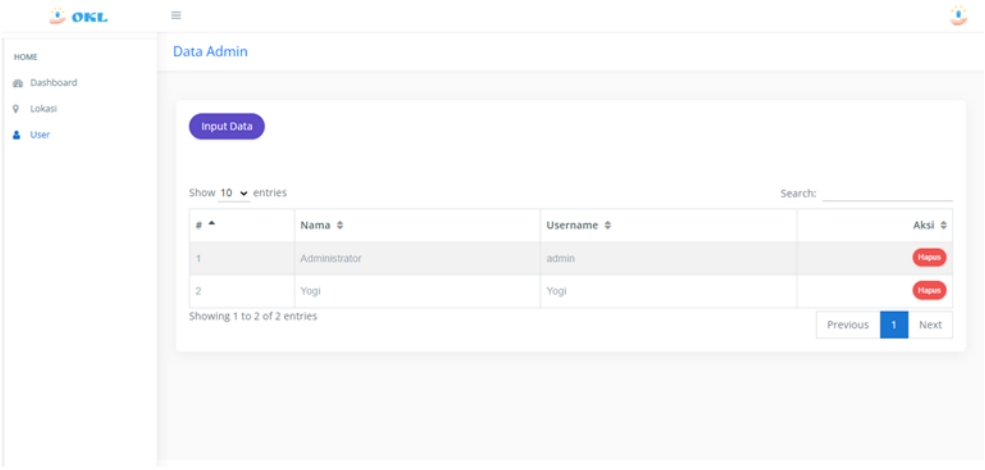
Hasil pengujian menggunakan metode pengujian *black-box*. Pengujian *black-box* terdiri dari lima komponen, yaitu pengujian fungsi menu dan tombol, pengujian antarmuka, pengujian kinerja dan perilaku pemuatan, pengujian struktur dan *database*, serta pengujian inisiasi dan terminasi. Pengujian aplikasi dilakukan pada tiga perangkat dengan spesifikasi dan ukuran layar berbeda.

3.1.1 Struktur pengujian dan *database*

Struktur datanya sesuai dengan *database* saat ini, dimana sistem dapat menampilkan nama pengguna, alamat, foto, garis bujur, lintang, dan informasi lainnya sehingga dapat ditunjukkan bahwa struktur *database* sudah sesuai dengan desain hubungan tersebut.

3.1.2 Inisiasi dan terminasi pengujian

Sistem dapat memulai proses login yang sesuai dan membuat data sesuai dengan pengguna yang *login*. Selain itu, *logout* sistem juga berjalan sesuai rencana karena langsung menghapus seluruh sesi yang ada.



Gambar 5. *Dashboard User* dan Admin.

3.2. Hasil Tes Jarak

Pengujian jarak diperlukan dalam penelitian ini untuk mengetahui keakuratan penghitungan jarak dengan menggunakan metode Haversine. Pengujian dilakukan dengan membandingkan perhitungan jarak antara perhitungan Haversine dengan perhitungan yang disediakan oleh aplikasi Google Maps. Hasil pengujian dapat dilihat pada Tabel 1.

Tabel 1. Hasil Pengujian.

ROUTE	METHODS	POINTS										ACCURACY
		1	2	3	4	5	6	7	8	9	10	
A-B	HAVERSINE	400	420	430	440	455	460	465	470	475	500	89%
	GOOGLE MAPS	445	465	475	485	500	505	510	515	520	545	
	DIFFERENT	45	45	45	45	45	45	45	45	45	45	
B-C	HAVERSINE	545	600	655	710	765	820	875	930	985	1,040	85%
	GOOGLE MAPS	600	655	710	765	820	875	930	985	1,040	1,095	
	DIFFERENT	55	55	55	55	55	55	55	55	55	55	
C-D	HAVERSINE	550	600	650	700	750	800	850	900	950	1,000	84%
	GOOGLE MAPS	600	650	700	750	800	850	900	950	1,000	1,050	
	DIFFERENT	50	50	50	50	50	50	50	50	50	50	
D-E	HAVERSINE	300	355	410	465	520	575	630	685	740	795	90%
	GOOGLE MAPS	355	410	465	520	575	630	685	740	795	850	
	DIFFERENT	55	55	55	55	55	55	55	55	55	55	
E-F	HAVERSINE	625	665	705	745	785	825	865	905	945	985	79%
	GOOGLE MAPS	665	705	745	785	825	865	905	945	985	1,025	
	DIFFERENT	40	40	40	40	40	40	40	40	40	40	
F-G	HAVERSINE	700	750	800	850	900	950	1,000	1,050	1,100	1,150	81%
	GOOGLE MAPS	750	800	850	900	950	1,000	1,050	1,100	1,150	1,200	
	DIFFERENT	50	50	50	50	50	50	50	50	50	50	
G-H	HAVERSINE	550	595	640	685	730	775	820	865	910	955	83%
	GOOGLE MAPS	595	640	685	730	775	820	865	910	955	1,000	
	DIFFERENT	45	45	45	45	45	45	45	45	45	45	
H-I	HAVERSINE	600	650	700	750	800	850	900	950	1,000	1,050	83%
	GOOGLE MAPS	650	700	750	800	850	900	950	1,000	1,050	1,100	
	DIFFERENT	50	50	50	50	50	50	50	50	50	50	

Tabel 1 menyajikan hasil perbandingan antara algoritma Haversine dan Google Maps untuk menghitung jarak rute yang relevan dengan aplikasi pengumpulan sampah. Data menunjukkan bahwa algoritma Haversine mencapai tingkat akurasi yang dapat diterima untuk tujuan ini. Dari delapan perbandingan rute (A-B hingga H-I), algoritma Haversine menghasilkan akurasi rata-rata 84% dibandingkan Google Maps. Rata-rata deviasi kedua metode tersebut adalah 112,522 meter.

Algoritma Haversine adalah persamaan matematika yang digunakan untuk menghitung jarak lingkaran besar antara dua titik pada sebuah bola. Dalam konteks aplikasi pengumpulan sampah, persamaan ini dapat digunakan untuk memperkirakan jarak antara lokasi pembuangan sampah dan titik pengumpulan. Meskipun Google Maps kemungkinan besar menggunakan serangkaian algoritma yang lebih canggih untuk memperhitungkan jaringan jalan raya dan kompleksitas geografis lainnya, algoritma Haversine menawarkan alternatif yang lebih sederhana dan efisien secara komputasi. Eksperimen pada riset ini menunjukkan bahwa algoritma Haversine memberikan tingkat akurasi yang sesuai untuk perencanaan rute pengumpulan sampah, dengan rata-rata deviasi kurang dari 150 meter.

Keakuratan algoritma Haversine dapat bervariasi tergantung pada wilayah geografis tertentu dan distribusi titik data. Untuk aplikasi yang memerlukan presisi tinggi, berkonsultasi dengan Google Maps atau layanan geospasial lainnya mungkin lebih baik. Pertukaran antara efisiensi komputasi dan akurasi harus dipertimbangkan ketika memilih metode penghitungan jarak untuk aplikasi pengumpulan sampah.

4. KESIMPULAN

Penelitian ini telah mengembangkan aplikasi *mobile* berbasis Android yang menggunakan data geospasial dan algoritme Haversine untuk mengoptimalkan efisiensi pembuangan sampah. Dengan peningkatan signifikan dalam jumlah sampah nasional, aplikasi ini menggunakan *geo-tagging* untuk menentukan lokasi pembuangan sampah terdekat, yang diharapkan dapat memperlancar proses pengumpulan sampah bagi petugas kebersihan. Penggunaan algoritma Haversine membantu aplikasi mencapai tingkat akurasi 84% dalam menghitung jarak dengan deviasi rata-rata 112,522 meter. Pengujian *black-box* memastikan fungsionalitas antarmuka aplikasi, dan optimasi sumber data diupayakan untuk meningkatkan kecepatan aplikasi dan perencanaan rute. Penelitian ini menegaskan bahwa integrasi algoritma Haversine dengan teknologi *geo-tagging* dapat meningkatkan akurasi dan efisiensi perencanaan rute pengumpulan sampah. Dengan menggunakan teknologi ini, aplikasi *mobile* Android menawarkan alternatif yang efisien secara komputasi dan dapat diterapkan untuk meningkatkan praktik pengelolaan sampah yang berkelanjutan, khususnya di Indonesia.

DAFTAR PUSTAKA

[1] M. J. Ismail, Pendidikan Karakter Peduli Lingkungan dan Menjaga Kebersihan di Sekolah, *Guru Tua: Jurnal Pendidikan dan Pembelajaran*, Vol. 4, No. 1, pp. 58-69, 2021, doi: 10.31970/gurutua.v4i1.67

[2] S. Hidayatuloh & N. S. Pratami, Rancang Bangun Sistem Transaksi Tabungan Untuk Pengelolaan Sampah Berbasis Web (Studi Kasus: Bank Sampah Sahitya Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta, *TEKINFO*, Vol. 22, No. 2, pp. 87-108, 2021.

[3] E. Winarno, W. Hadikurniawati, & R. N. Rosso, Location Based Service for Presence System Using Haversine Method, *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, Salatiga, Indonesia, pp. 1-4, 2017, doi: 10.1109/INNOCIT.2017.8319153.

[4] A. R. F. Rabbani & A. R. Pratama, Aplikasi Sistem Jemput Sampah Berbasis Android untuk Rumah Kos dan Area Sekitar Kampus, *Jurnal. Sains dan Informatika*, Vol. 7, No. 1, pp. 67–76, 2021, doi: 10.34128/jsi.v7i1.299.

[5] I. Yang, W. H. Jeon, & J. Moon, A Study on a Distance- Based Coordinate Calculation Method Using Inverse Haversine Method, *Journal of Digital Contents Society*, Vol. 20, No. 10, pp. 2097–2102, 2019, doi: 10.9728/dcs.2019.20.10.2097.

[6] D. Ikasari, W. Ikasari, & R. Andika, Implementation of Haversine Formula to Determine the Shortest Path Using Web Based Application for a Case Study of High School Zoning in Depok, *American Journal of Software Engineering and Applications*, Vol. 10, No. 2, pp. 19-31, 2021, doi: 10.11648/j.ajsea.20211002.11.

[7] D. A. Prasetya, P. T. Nguyen, R. Faizullin, I. Iswanto, & E. F. Armay, Resolving the Shortest Path Problem Using the Haversine Algorithm, *Journal of Critical Reviews*, Vol. 7, No. 1, pp. 62– 64, 2020, doi: 10.22159/jcr.07.01.11.

[8] R. N. Yuniar, P. A. Kencan, E. Ruth, & A. H. S. Budi, Analysis of Estimated Busses Arrival Time on Public Transportation Using Real-Time Monitoring, *IOP Conf. Series: Materials Science and Engineering*, 850 012026, 2020, doi: 10.1088/1757-899X/850/1/012026.

[9] A. Rahimi, T. Cohn, & T. Baldwin, Pigeo: A Python Geotagging Tool, *Proceedings of ACL-2016*

System Demonstrations, Berlin, Jerman, pp. 127–132, 2016, doi: 10.18653/v1/p16-4022.

[10] H. Alkan & H. Celebi, The Implementation of Positioning System with Trilateration of Haversine Distance, *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Istanbul, Turki, pp. 1–6, 2019, doi: 10.1109/PIMRC.2019.8904289.

[11] I. Setyorini & D. Ramayanti, Finding Nearest Mosque Using Haversine Formula on Android Platform, *Jurnal Online Informatika*, Vol. 4, No. 1, p. 57-62, 2019, doi: 10.15575/join.v4i1.267.

[12] R. S. Abhi Krishna & S. Ashok, Automated Land Area Estimation for Surveying Applications, *2020 International Conference for Emerging Technology (INCET)*, Belgaum, India, pp. 1–5, 2020, doi: 10.1109/INCET49848.2020.9154042.

[13] G. T. S. Lee, D. Arisandi, & Wasino, Travel App - Showing Nearest Tourism Site Using Haversine Formula and Directions with Google Maps, *IOP Conf. Series: Materials Science and Engineering*, 852 012161, 2020, doi: 10.1088/1757-899X/852/1/012161.