Journal of Universal Computer Science, vol. XX, no. X1 (2023), XXXX-XXXX
submitted: 1/1/2021, accepted: 2/2/2021, appeared: 3/3/2021 CC BY-ND 4.0

ANDROID GEOTAGGING HAVERSINE ALGORITHM MODEL FOR NEARBY ENVIRONMENTAL CLEANING ROUTE SEARCH

Riko Herwanto
Institut Informatika dan Bisnis Darmajaya, Bandar Lampung, Lampung, Indonesia
https://orcid.org/ 0000-0001-7452-5107 , rikoherwanto@darmajaya.ac.id

Nisar Zaidal
Institut Informatika dan Bisnis Darmajaya, Bandar Lampung, Lampung, Indonesia
nisar@darmajaya.ac.id

Chairani Fauzi
Institut Informatika dan Bisnis Darmajaya, Bandar Lampung, Lampung, Indonesia
chairani@darmajaya.ac.id

Muhammad Yogi
Institut Informatika dan Bisnis Darmajaya, Bandar Lampung, Lampung, Indonesia
1911911010156.1911010156@mail.darmajaya.ac.id

Abstract: The environment is a global problem due to too many people, insufficient natural resources, and careless use of nature. Garbage is the leftover stuff that people throw away, and Indonesia's total waste was 65.8 million tons in 2017 and 752 million tons in 2018. This research uses the Haversine algorithm and geotagging to create an app that makes it easier to find trash cans based on the closest route. Distance testing is required to determine the accuracy of calculating distances, and interface testing results use the black box testing method. Data-source direct roles can make applications faster and make it easier for the user to determine the place waste desired by environmental cleaners. The distance calculation using the haversine method is 84% of the actual distance, and the average difference is 112.522 meters.

Keywords: Geo-Tagging, Haversine, Mobile, Sort-Route,
Categories: D.3.3, F.2.1, G.1.6, H.1.1, H.2.3, H.2.8, L.3.1.

1 Introduction

The environment is now a global problem that everyone cares about. The environment is worsening because there are too many people, insufficient natural resources, and careless use of nature. People's careless actions toward the environment are at the root of many of the problems in the global and national environments. This is based on many years of research and observation. Most environmental damage is caused by people's actions that put humans outside nature and nature at the center of the natural system. [1].

The higher the level of public consumption, the faster the economy grows. People do not just use and consume natural resources to live and meet their needs anymore; they do more than that. This means that there are a lot of unused items that can hurt the environment. Garbage is the leftover stuff people throw away, so the problem of garbage is often spread around carelessly. Carelessly throwing away trash can endanger people's lives in many ways, like by causing flooding, a dirty environment, air pollution, and so on. [2]. According to our observations and conversations with other officers, the primary issue with environmental cleanliness is that individuals do not comprehend why environmental cleaning activities are necessary. The location of the trash can is based on the route of the environmental cleaner, which is the closest route[3]. Officials in charge of waste need to know where waste is based on the closest route, but environmental officers do not have this information. Based on the problems that environmental officers face, the search method can be used to find the closest route for cleaning up the environment through the use of the Haversine algorithm and geotagging[4]. The Haversine algorithm is used in applications that give information about shortest-distance navigation. Geotagging adds data position information to GPS in the form of latitude and longitude information in an image. Here are some benefits of this research:

Environmental cleaners want to make an app that makes it easier to find trash cans based on the closest route. Helping people who clean up the environment find where trash is based on the shortest route the cleaner wants

2 Literarure Review

2.1 Haversine Algorithm

Haversine is an integral equation in navigation systems [5]. Later, the Haversine formula calculates the shortest distance between two points, such as longitude and latitude, on a ball [6]. Haversine is an application of the concept of trigonometry, which is part of geometry. The Haversine formula is an essential navigational equation that shows the great circle distance between two points (latitude and longitude) on the surface of a sphere (the earth) as a function of longitude and latitude. The use of this method is correct, disregarding the surface height of hills and the depth of valleys[7].

Haversine is an integral equation in navigation systems. Later, the Haversine formula calculates the shortest distance between two points, such as longitude and latitude, on a ball. Haversine is an application of the concept of trigonometry, which is part of geometry. The Haversine formula is an essential navigational equation that shows the great circle distance between two points (latitude and longitude) on the surface of a sphere (the earth) as a function of longitude and latitude. Although the method does not take into account the topography (i.e., the height of hills and the depth of valleys), it is nevertheless entirely accurate[8].

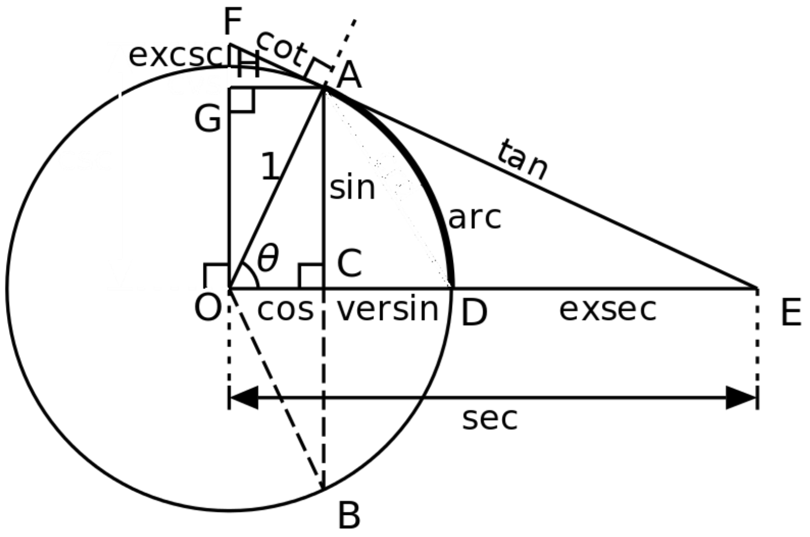


Figure 2. 1 The Haversine Pattern

Figure 2.1 illustrates the Haversine formula pattern described in spherical trigonometry. Where this equation is fundamental in navigation, this Haversine formula will produce the shortest distance between two points. This formula was originally used for the main problems of nautical astronomy. Harvesine is used to determine the distance between stars[8]. First used by Josef de Mendoza y Rios in 1801, and Jamez Andrew discovered this formula in 1805. The term "haversine" itself was coined or named in 1835 by Prof. James Inman. Assuming that the earth is perfectly spherical with a radius of 6.3671 km and that the locations of the two points on the spherical coordinates (latitude and longitude) are lon1, lat1, and lon2, lat2, the Haversine formula can be written as follows:

The Haversine formula is the first equation to consider when calculating distances on a sphere[9]. The word "Haversine" comes from the function:

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

The following equation where φ is latitude, λ is longitude, and R is earth's radius (mean radius = 6,371km) is how we translate the above formula to include latitude and longitude coordinates. Note that angles need to be in radians to pass to trig functions:

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

$$c = 2 \times \operatorname{atan} 2\ (\sqrt{a, \sqrt{(1-a)})} \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$d = R \times c \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

2.2 Geo-Tagging

(Wong, Law, and Li 2017)) say that "geotagging," which is also called "geo-referencing," is the process of adding metadata to images and videos that tells where they were taken. This process can help users find various kinds of information about a location. (Mandal et al. 2022), manual geotagging is a method where information about the location is added manually by inputting specific coordinates or selecting a location when uploading media to the internet. The level of accuracy of this geotagging method depends on the tools used or the GPS receiver to obtain accurate coordinates (Basyir et al. 2018)

Google Map API is a tool or service provider provided by a technology called Google to its users so they can take advantage of Google Maps when developing an application being built(Alkan and Celebi 2019). The Google Maps API also has different ways to get data and add contacts through different service providers, and it lets more than one user access the data (Lee, Arisandi, and Wasino 2020). Build enterprise systems within applications.

3 Research Methodology

Research design is a concept or image of the research to be carried out. The description of the research flow can be seen in Figure 3.

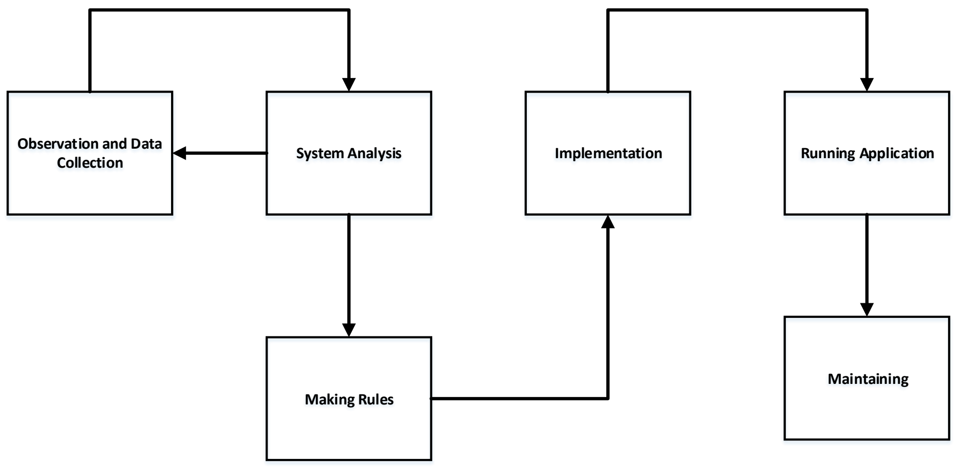


Figure 3.1. Research Design

Observation and Data Collection.

Observation and data collection are functions for data collection where the researcher records all information gathered throughout the research, which is a method of gathering data by direct observation of situations or occurrences in the field.

System Analysis

After consulting with users, the system's services, limitations, and goals are determined at this stage. Everything is defined in detail and made into a system specification from the observations and data collection done by the researcher.

Making Rules.

The rule-making process provides the hardware or software requirements by providing the entire system's architecture. System design involves identifying and describing system abstractions.

Implementation.

At this stage, system design is realized as a program or program unit. Unit testing involves verifying whether each unit meets system specifications.

Running Application

After the program units have been designed, they will be tested to run applications that involve verification to ensure each unit meets the system specifications.

Maintaining

Maintenance installed the system and used it in practice. Maintenance means fixing errors not found in the previous stage, improving the system unit's implementation, and improving the services the system offers as new needs are found.

Combining the Haversine Algorithm with Geotagging involves several steps:

1. Collecting geographical data: The first step is to collect geographical data, such as latitude and longitude coordinates, for each location that you want to include in your application or service. This data can be obtained from GPS devices, maps, or other sources[10].

2. Implementing the Haversine Algorithm: The next step is to implement the Haversine Algorithm in your application or service. The Haversine Algorithm uses the latitude and longitude coordinates to calculate the distance between two points on the surface of a sphere the code for the Haversine Algorithm is in your preferred programming language [11].

   Here is a simple Python implementation of the Haversine algorithm with geotagging:

```
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")
```

3. Storing geographical data: After collected the geographical data and implemented the Haversine Algorithm, the next step is to store the data in a database or other data storage system. Use a NoSQL database, such as MongoDB or Cassandra, or a relational database, such as MySQL or PostgreSQL, to store the data .

4. Adding Geo Tagging: The final step is to add Geo Tagging to your application or service. Use the latitude and longitude coordinates to add geographical identification metadata to various media such as photos, videos, and websites.

By following these steps, we can combine the Haversine Algorithm with Geo Tagging to create location-based services and applications that can accurately determine the distance between two points and associate that information with specific media or other data.

To calculate the accuracy of combining the Haversine Algorithm with Geo Tagging [12], we compare the results obtained from the combination to actual ground truth data. Here is a general outline of how you could do this:

1. Obtain ground truth data: This involves obtaining accurate, real-world data for the locations you want to compare with your results. We obtain this data through various means, such as field surveys, GPS measurements, or by using a known, accurate reference location.

2. Use the Haversine Algorithm with Geo Tagging: Calculate the distances between the locations using the Haversine Algorith (Placeholder2) m with the geo-tagged coordinates.

3. Compare the results with the ground truth data: Compare the results obtained from the Haversine Algorithm with the ground truth data. This can be done by calculating the difference between the two sets of data.

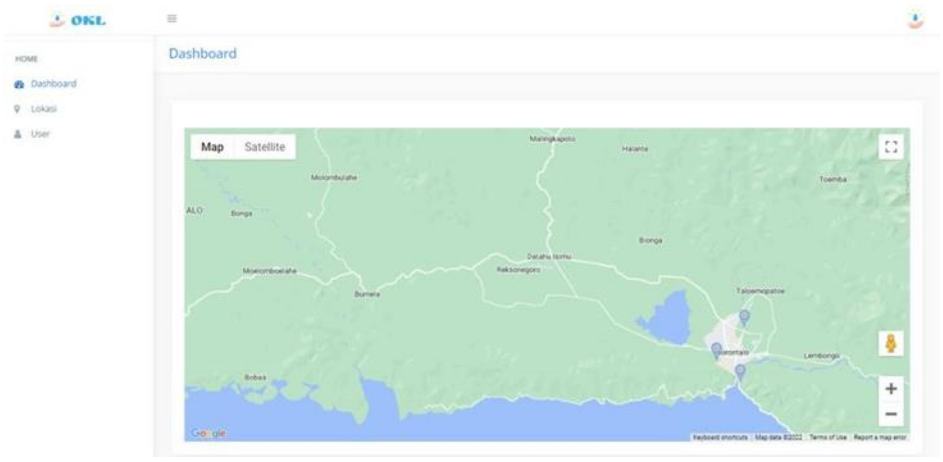
4. Calculate the accuracy: To calculate the accuracy, you can use a metric such as the Root Mean Squared Error (RMSE), which measures the average deviation of the results from the ground truth data. A low RMSE value indicates a high accuracy, while a high RMSE value indicates a low accuracy.

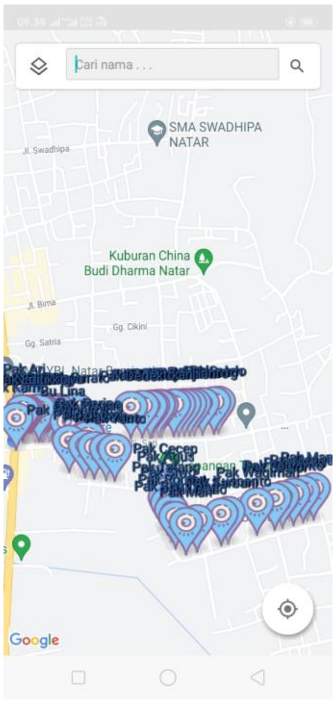
5. Repeat the process: Repeat the process multiple times with different sets of data to obtain an average accuracy value.

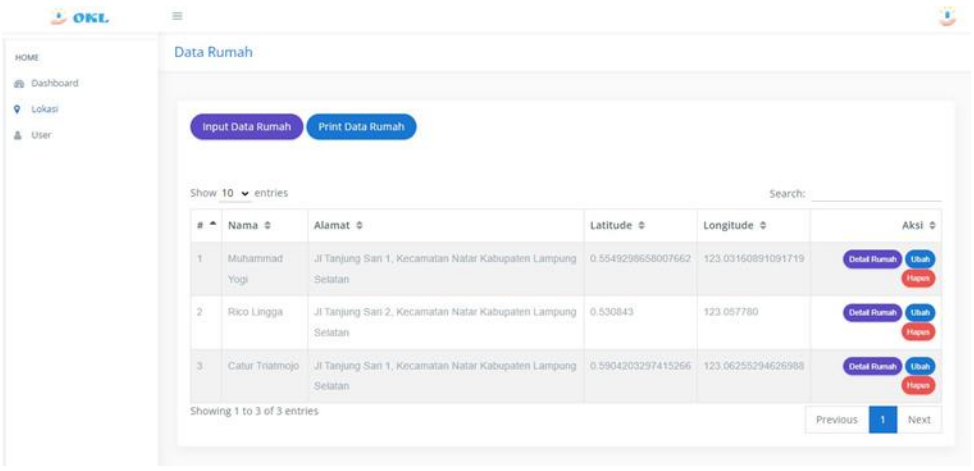
By following this process, you can calculate the accuracy of combining the Haversine Algorithm with Geo Tagging. The specific steps and details may vary depending on the data you have available and the specific use case you have in mind, but this provides a general idea of how you could calculate the accuracy of combining these two technologies [13].

4 Result

Distance testing is required in this study to determine the accuracy of calculating distances using the Haversine method. The test is carried out by comparing the calculation of the distance between the Haversine calculations and the calculations provided by the Google Maps application.







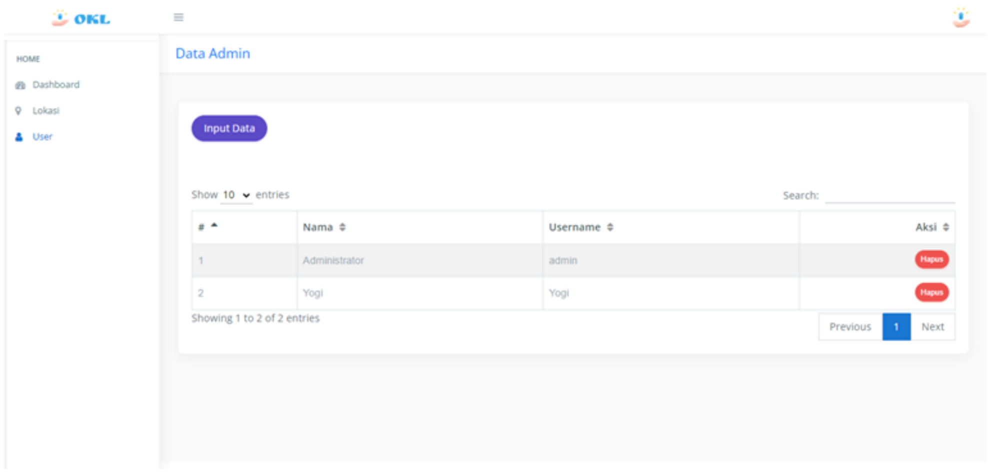
Data Rumah

Show 10 entries

Search:

#	Nama	Alamat	Latitude	Longitude	Aksi
1	Muhammad Yogi	Jl Tanjung Sari 1, Kecamatan Natar Kabupaten Lampung Selatan	0.5549298658007662	123.03160891091719	Detail Rumah Ubah Hapus
2	Rico Lingga	Jl Tanjung Sari 2, Kecamatan Natar Kabupaten Lampung Selatan	0.530843	123.057780	Detail Rumah Ubah Hapus
3	Catur Triatmojo	Jl Tanjung Sari 1, Kecamatan Natar Kabupaten Lampung Selatan	0.5904203297415266	123.06255294626988	Detail Rumah Ubah Hapus

Showing 1 to 3 of 3 entries



#	Nama	Username	Aksi
1	Administrator	admin	Hapus
2	Yogi	Yogi	Hapus

Showing 1 to 2 of 2 entries

Figure 4.1. Design Interface

Interface Testing Results

The test results use the black box testing method. Black box testing consists of five components: menu and button function tests, interface tests, loading performance and behavior tests, structure and database tests, and initiation and termination tests. Application testing is carried out with three devices with different specifications and screen sizes.

Test structure and database

The data structure matches the current database, where the system can show the user's name, address, photo, longitude, latitude, and other information so that it can be shown that the structure of the database is in line with that relationship design.

Test initiation and termination

The system can initiate the appropriate login process and create data according to the user logged in. In addition, logging out of the system also goes according to plan because it immediately deletes all existing sessions.

Results of the Distance Test

Distance testing is required in this study to determine the accuracy of calculating distances using the Haversine method. The test is carried out by comparing the calculation of the distance between the Haversine calculations and the calculations provided by the Google Maps application[14].

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

Table 1. Result Measuring

From the test above, it can be seen that the calculation using haversine has a low level of accuracy if the road is curved. The Google Maps application could also have a better accuracy value for distance calculations[15], even though the calculation has been done using calculations based on the path traversed.

5 Conclusions

Data-source direct roles can make applications faster. Develop and make it easier for the user to determine the place waste desired by environmental cleaners. In determining the position of the nearest garbage can, the haversine technique provides a more precise distance estimate than previous approaches. The distance is 84% of the actual distance, and the average difference is 112.522 meters.

Future Work

In near future, this research can be connected with RFID or IoT technology by forming a waste disposal management which is usually implemented in the concept of smart-village or smart-city.

Acknowledgements

This work would not have been possible without the support of the Darmajaya Institute of Informatics and Business. I am especially indebted to fellows lecturer at Department of Informatics Technology, who have been supportive of my goals and who worked actively to provide me with the protected academic time to pursue those goals.

References

[1] M. J. Ismail, "MENJAGA KEBERSIHAN DI SEKOLAH," vol. 4, no. 1.

[2] S. Hidayatuloh, N. S. Pratami, S. Informasi, and T. Selatan, "RANCANG BANGUN SISTEM TRANSAKSI TABUNGAN UNTUK PENGELOLAAN SAMPAH BERBASIS WEB (STUDI KASUS : BANK SAMPAH SAHITYA FAKULTAS SAINS DAN TEKNOLOGI UIN SYARIF," vol. 22, no. 2.

[3] E. Winarno, W. Hadikurniawati, and R. N. Rosso, "Location based service for presence system using haversine method," *Proc. - 2017 Int. Conf. Innov. Creat. Inf. Technol. Comput. Intell. IoT, ICITech 2017*, vol. 2018-Janua, pp. 1–4, 2018, doi: 10.1109/INNOCIT.2017.8319153.

[4] A. R. F. Rabbani and A. R. Pratama, "Aplikasi Sistem Jemput Sampah Berbasis Android untuk Rumah Kos dan Area Sekitar Kampus," *J. Sains dan Inform.*, vol. 7, no. 1, pp. 67–76, 2021, doi: 10.34128/jsi.v7i1.299.

[5] M. Furqan, "MFQ- 2017-2," 2017.

[6] I. Yang, W. H. Jeon, and J. Moon, "A study on a distance based coordinate calculation method using Inverse Haversine Method," *J. Digit. Contents Soc.*, vol. 20, no. 10, pp. 2097–2102, 2019, doi: 10.9728/dcs.2019.20.10.2097.

[7] D. Ikasari, W. Ikasari, and 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," *Am. J. Softw. Eng. Appl.*, vol. 10, no. 2, p. 19, 2021, doi: 10.11648/j.ajsea.20211002.11.

[8] T. C. Formula, "[January," vol. 64, no. 1, pp. 38–40, 2015.

[9] D. A. Prasetya, P. T. Nguyen, R. Faizullin, I. Iswanto, and E. F. Armay, "Resolving the shortest path problem using the haversine algorithm," *J. Crit. Rev.*, vol. 7, no. 1, pp. 62–64, 2020, doi: 10.22159/jcr.07.01.11.

[10] R. N. Yuniar, P. A. Kencan, E. Ruth, and A. H. S. Budi, "Analysis of estimated busses arrival time on public transportation using real-time monitoring," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 850, no. 1, 2020, doi: 10.1088/1757-899X/850/1/012026.

[11] A. Rahimi, T. Cohn, and T. Baldwin, "Pigeo: A Python geotagging tool," *54th Annu. Meet. Assoc. Comput. Linguist. ACL 2016 - Syst. Demonstr.*, no. 4, pp. 127–132, 2016, doi: 10.18653/v1/p16-4022.

[12] H. Alkan and H. Celebi, "The Implementation of Positioning System with Trilateration of Haversine Distance," *IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC*, vol. 2019-Septe, pp. 1–6, 2019, doi: 10.1109/PIMRC.2019.8904289.

[13] I. Setyorini and D. Ramayanti, "Finding Nearest Mosque Using Haversine Formula on Android Platform," *J. Online Inform.*, vol. 4, no. 1, p. 57, 2019, doi: 10.15575/join.v4i1.267.

[14] R. S. Abhi Krishna and S. Ashok, “Automated land area estimation for surveying applications,” *2020 Int. Conf. Emerg. Technol. INCET 2020*, pp. 1–5, 2020, doi: 10.1109/INCET49848.2020.9154042.

[15] G. T. S. Lee, D. Arisandi, and Wasino, “Travel App - showing nearest tourism site using Haversine formula and directions with Google Maps,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 852, no. 1, 2020, doi: 10.1088/1757-899X/852/1/012161.