Journal Sensi

Print ISSN: 2461-1409
Online ISSN: 2655-5298

Research Article **Open Access (CC–BY-SA)**

The Use of Geofencing in Android-Based Mobile Applications for Promotional Ads in Shopping Centers

Yodi Susanto*[1], Persis Haryo Winasis[2], Muhammad Maulana Rachman[3], Heriyanto[4]
*[1]*Teknik Informatika, IPWIJA University, Jakarta*
[2,3]*Rekayasa Perangkat Lunak, IPWIJA University, Jakarta*
[4]*Pendidikan Keagamaan Buddha, STABN Sriwijaya, Tangerang*
*e-mail: *[1]yodisusanto@ipwija.ac.id, [2]persisharyo@ipwija.ac.id, [3]maulana@ipwija.ac.id, [4]heriyanto@stabn-sriwijaya.ac.id*

Abstract

This paper describes the implementation of geofencing technology in Android-based mobile applications for promotional advertisements based on the user's location. Geofencing is a location-based service that uses GPS, cellular networks, or Wi-Fi to create virtual geographic boundaries around specific locations. This technology enables businesses to send targeted advertisements to mobile users based on their current location. The paper presents the development of an Android-based mobile application that uses geofencing technology to provide location-based promotional advertisements to users. Researchers use one of the big shopping centers in Jakarta as a location object for geofencing technology. The application uses Google Maps and Firebase to create geofences around specific locations, and then sends push notifications to users within the geofenced areas. The implementation also includes a backend system to manage the promotional advertisements and user data. The results show that the implementation of geofencing technology in Android-based mobile applications can provide a more targeted and effective way to deliver promotional advertisements to users.

Keywords**—***geofencing, geofence, mobile applications, android, promotional advertisements, targeted advertisements*

1. INTRODUCTION

In the current digital era, technology has had a significant impact on the way people interact and shop. The use of smartphones has increased and has become a daily necessity in people's lives. This has resulted in the increased use of mobile applications that make it easier for consumers to make purchases and find product information. One of the complete shopping destinations is a shopping center. Shopping centers are considered strategic places for advertisers to promote their products because they are crowded with visitors. It takes targeted promotional media to increase consumer shopping traffic in a shopping center.

Geofencing is a feature that can be developed in software that uses the Global Positioning System (GPS) or radio frequency identification (RFID) to determine a geographic boundary and utilizes location information to build a better user experience [1]. Geofence empowers geographic observation of the area covered by the virtual boundary area [2]. This technology allows users to map locations and send specific information tailored to the mapped location [3]. Therefore, geofencing can be used as an effective tool for promoting products in shopping centers. With the help of location-based services, businesses can deliver personalized and relevant advertisements to mobile users based on their current location. The technology enables businesses to send targeted advertisements to users within the geofenced areas [1]. The objectives of this research are to understand and develop Android-based mobile applications with geofencing technology to promote promotional advertisements in shopping centers.

Furthermore, this research aims to increase the effectiveness of promotional advertisements in shopping centers and make it easier for users to find and purchase the products they want.

2. RESEARCH METHOD

1.1. Method of Collecting Data

Researchers collected the necessary data related to the use of geofences in mobile applications and their usefulness in distributing advertisements to shopping centers. The data collection methods used are as follows.

a. Observation Method
Observation is the process of obtaining information by keeping an eye on actions, occurrences, or physical traits in their natural environment [2]. Researchers conducted conservation in a shopping center to see the density of visitors, consumer habits in finding information about products in the shopping center area, coverage area for GSM signals, and free internet access provided by the shopping center management.
b. Interview Method
In qualitative research designs, interviews are the cornerstone of the primary data collection process [3]. Researchers conducted interviews with several related parties to obtain information in accordance with real-life conditions in the field. Interviews were conducted with the management of the shopping center, namely the customer service division, marketing and communication division, and operational division. Researchers also conducted interviews with several consumers with an age range of 20–40 years.
c. Literature Study Method
No matter the discipline, the foundation of all academic research efforts is building on prior research and connecting it to information already in existence. Therefore, accuracy in doing so should to be a top priority for all academics [4]. The method of spending literature is carried out to support the interview method and observation method. This approach gathers data from the written word and the internet by reading and researching a number of publications and books on research theories relevant to this study.

1.2. System Design Method

Modeling is created by preparing class diagrams, use case diagrams and mockups of user interfaces to be built. In this research will be built mobile apps with an android base. The programming language uses java (Android Native) with the Android Studio version 4.1.3 IDE (Integrated Development Environment). For web services using php 7 programming language with Codeigniter 4 framework. The Tools used is Sublime Text, while for databases it uses Microsoft SQL Server 2012.

1.3. Testing Method

The design testing phase of the resulting program is carried out in this study to ensure that the program can run in accordance with predetermined success indicators [5]. After the system testing and integration testing phases have been completed, acceptance testing is the last phase of functional testing and is used to assess whether or not the final piece of software is ready for delivery. It involves ensuring that the product is in compliance and meets the end user's needs.

1.4. Previous Research

The use of geofences has been very helpful in providing solutions in several fields. Researchers use references from several previous studies that discuss geofencing and other related fields. The research that is used as a reference can be seen in the following Table 1.

Table 1. Previous Research

Author (years)	Title
Marti Widya Sari, Banu Santoso, Mohamed Nor Azhari Azman. (2021) [6]	Implementing Geo Positioning System for Children Tracking Location Monitoring based on Android
Alya Geogiana Buja, Noor Afni Deraman, Khyrina Airin Fariza Abu Samah, Mohd Nor Hajar Hasrol Jono, Mohd Ali Mohd Isa and Shahadan Bin Saad. (2021) [7]	Enhancing a mobile application with contextual information delivery using proximity beacon: a preliminary study at a tourist destination
Amadiea permana Sanusi, Aad Hariyadi, M. Nanak Zakaria. (2020) [8]	Implementasi Teknologi Geofencing Untuk Pengawasan terhadap Lansia Menggunakan Sarung Lengan Berbasis Mikrokontroler dan Android
M. Makhtar, R. Rosly, S. A. Fadzli, S. N. W. Shamsuddin and A. A. Jamal. (2016) [9]	Implementation Of Mobile Attendance Application Using Geo-Fence Technique
Sandro Rodriguez Garzon, Bersant Deva, Gabriel Pilz, Stefan Medack. (2015) [10]	Infrastructure-assisted Geofencing: Proactive Location-based Services with Thin Mobile Clients and Smart Servers

3. RESULTS AND DISCUSSION

3.1. Requirements Definition

The analysis is carried out on the needs of the entire system to be built, because the software application will interact with various other elements. The author uses the GeofencingApi library from Google Play Services, because it is easier to use and has clear documentation. In the library some classes that we can use include Geofence, GeofencingRequest, GeofencingEvent and GeofenceStatusCodes.

3.2. System and Software Design

At this stage, the design and modeling of data structures, program algorithms, and software interfaces will be carried out. Then the results of the documentation will be used by programmers in creating software applications by implementing geofence technology.

3.2.1 Class Diagram

The class diagram applied in making mobile applications in shopping centers on this occasion can be seen in figure 1 below.

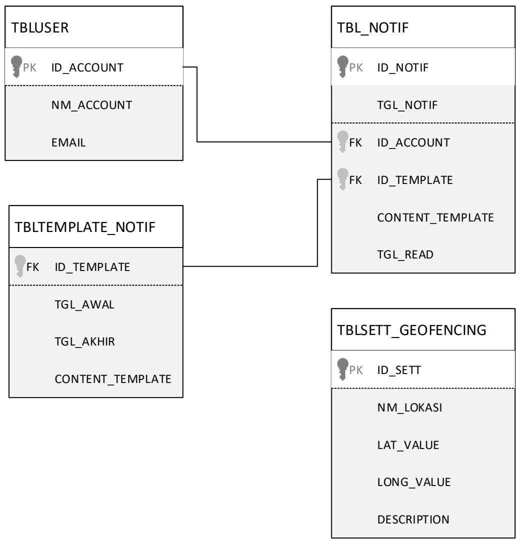


Figure 1. Class diagram

3.2.2 Activity Diagram

An activity diagram is essentially a flowchart that shows the activities performed by a system. In this research, an activity diagram was created with a focus on the process of implementing geofencing in mobile applications. The process takes place when visitors enter a predetermined geofence area. The activity diagram can be seen in figure 2.

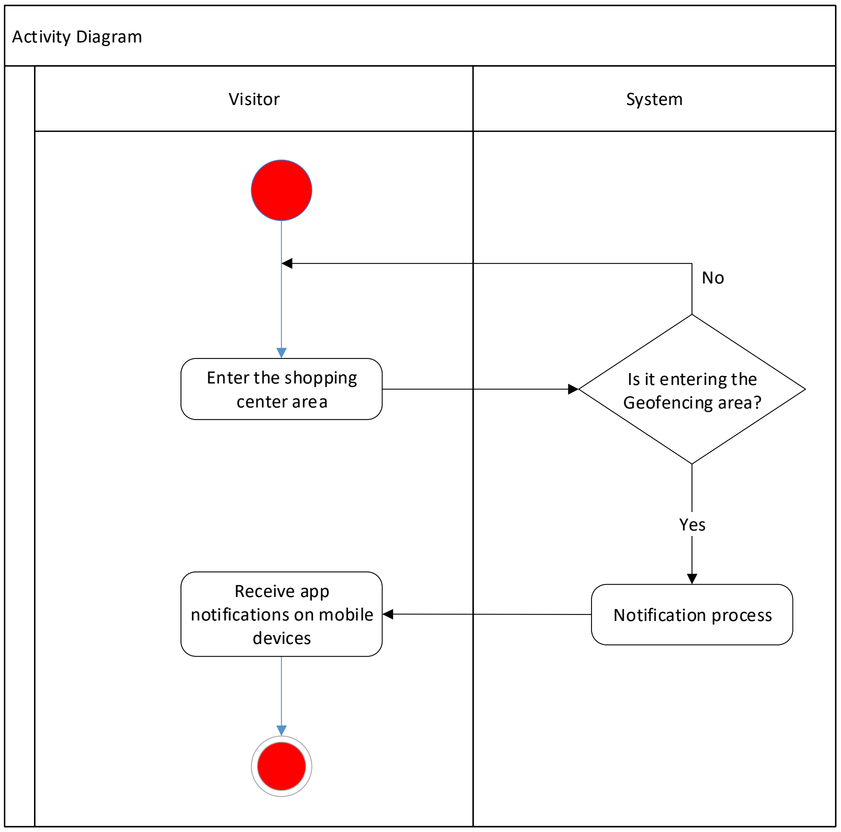
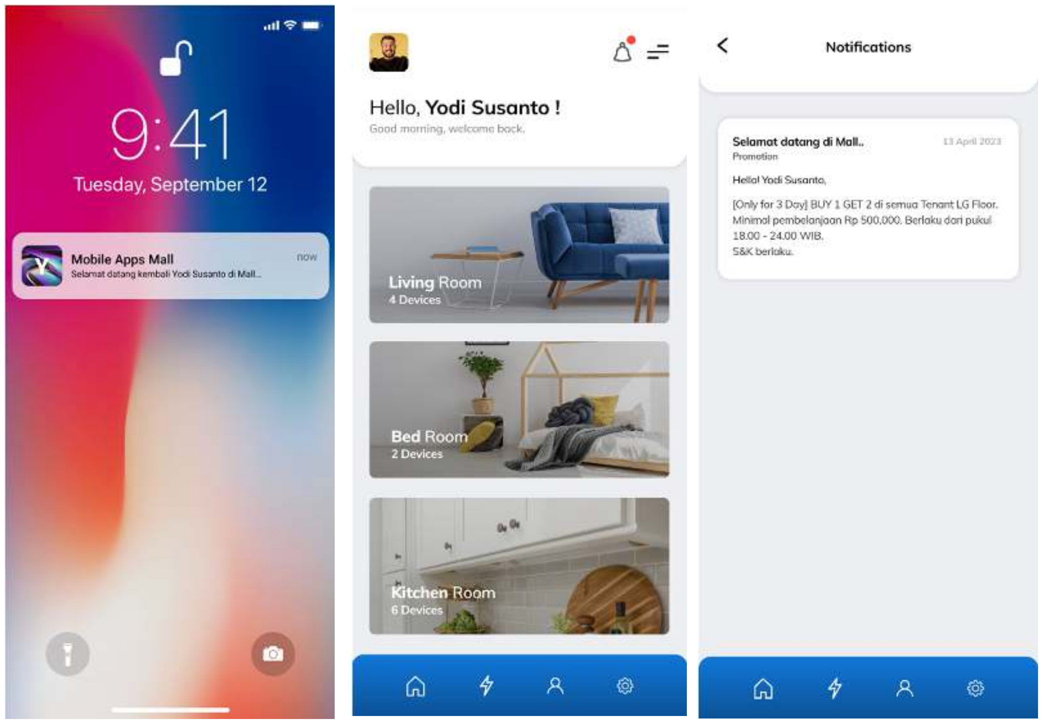

Activity Diagram
Visitor
System
No
Enter the shopping center area
Is it entering the Geofencing area?
Yes
Receive app notifications on mobile devices
Notification process

Figure 2. Activity diagram

3.2.3 User Interface

This interface is one of the menus of a membership application in a shopping center. In the backend system, we have determined which parameter areas will be included in the geofence coverage area. When users enter the shopping center area in a geofencing area, the system will provide notifications to them through mobile apps. Then, when the notification is opened, the mobile application will provide information on promotions and events that are being held in the area.

Shopping center managers can attract the attention of visitors by providing attractive notifications on their mobile devices. By displaying attractive product images on the application, visitors' shopping interests can be directed to certain promotions. In figure 3, you can see the interface of the application applied to members and visitors to the shopping center.



Mobile Apps Mall
now
Selamat datang kembali Yodi Susanto di Mall...

Selamat datang di Mall..
13 April 2023
Promotion

Hello! Yodi Susanto,

[Only for 3 Day] BUY 1 GET 2 di semua Tenant LG Floor. Minimal pembelanjaan Rp 500.000. Berlaku dari pukul 18.00 - 24.00 WIB.
S&K berlaku.

Figure 3. User interface

3.3. Implementation

This stage involves the creation of an application using the programming code specified in the software requirements document. At this stage, the creation of a database and infrastructure that support the system's ability to work well is also carried out. The result at this stage is a beta version of the software.

Geofence is a representative of geographical areas. In this class, we will register the geographical area to be monitored by providing latitude and longitude area coordinates, the radius of the area in meters, determine the time for how long the geofencing is active, and finally determine the geofence transition, namely the action that will be monitored in the registered area. In this geofence discussion, there are three types of transitions, as illustrated in figure 4.

a. GEOFENCE_TRANSITION_DWELL
Transition indicates the user logged in and spent a certain amount of time within the geofencing area. This is useful to avoid many warnings when users log out and log in for a fast period of time, if using this property, add the setLoiteringDelay parameter.
b. GEOFENCE_TRANSITION_ENTER
Transition indicating the user is logged in to the geofencing area.
c. GEOFENCE_TRANSITION_EXIT
Transition indicating the user exited the geofencing area.

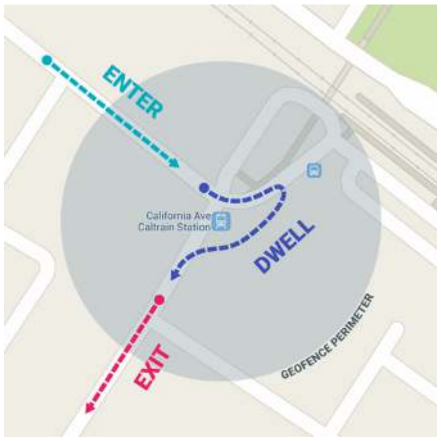


Figure 4. Geofence transition

If it has been registered, then the geofence class will be monitored by the geofencer service and will provide a warning if there are users entering or leaving the marked area, and we can perform certain actions as needed. An example of using the geofence class can be seen in figure 5.

```
Geofence geofence = new Geofence.Builder()
.setRequestId(GEOFENCE_REQ_ID) // Geofence ID
.setCircularRegion( LATITUDE, LONGITUDE, RADIUS) // defining fence region
.setExpirationDuration( DURANTION ) // expiring date
// Transition types that it should look for
.setTransitionTypes( Geofence.GEOFENCE_TRANSITION_ENTER | Geofence.GEOFENCE_TRANSITION_EXIT )
.build();
```

Figure 5. Geofence class

GeofencingRequest is a class that is used as a liaison to monitor from a customized area of the Geofence Class. We can create an instance of the GeofencingRequest Class using Builder(), then add a pre-customized Geofence and specify the notification type event when the Geofence is created. An example of using GeofencingRequest can be seen in the figure 6.

```
GeofencingRequest request = new GeofencingRequest.Builder()
// Notification to trigger when the Geofence is created
.setInitialTrigger( GeofencingRequest.INITIAL_TRIGGER_ENTER )
.addGeofence( geofence ) // add a Geofence
.build();
```

Figure 6. Geofencing request

GeofencingEvent is a class used to identify events to be handled by Geofencing sourced from any geofence transitions that have been determined at the beginning. An example of using GeofencingEvent can be seen in figure 7.

```java
GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);
int geoFenceTransition = geofencingEvent.getGeofenceTransition();
// Check if the transition type
if ( geoFenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
        geoFenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT ) {
    // Get the geofence that were triggered
    List<Geofence> triggeringGeofences = geofencingEvent.getTriggeringGeofences();
    // Create a detail message with Geofences received
    String geofenceTransitionDetails = getGeofenceTrasitionDetails(geoFenceTransition, triggeringGeofences );
    // Send notification details as a String
    sendNotification( geofenceTransitionDetails );
}
```

Figure 7. Geofencing event

GeofenceStatusCodes are classes that contain constant value information in the geofencing process. One example of the GeofenceStatusCodes class is to find out if geofencing is unavailable by using the value of the GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE constant. An example of using the GeofenceStatusCodes class can be seen in figure 8.

```java
public static String getErrorString(Context context, int errorCode) {
    Resources mResources = context.getResources();
    switch (errorCode) {
        case GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE:
            return mResources.getString(R.string.geofence_not_available);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_GEOFENCES:
            return mResources.getString(R.string.geofence_too_many_geofences);
        case GeofenceStatusCodes.GEOFENCE_TOO_MANY_PENDING_INTENTS:
            return mResources.getString(R.string.geofence_too_many_pending_intents);
        default:
            return mResources.getString(R.string.unknown_geofence_error);
    }
}
```

Figure 8. Geofence status code

3.3.1 Creating Geofencing Apps with android studio (Java)

The following are the stages of creating Geofencing Apps with android studio (Java).

a. Add library dependencies from google play service location to the build.gradle file as you can see in figure 9.

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Figure 9. Add library

b. Add permissions to AndroidManifest.xlm to access the location on the installed device as shown in figure 10.

```xml
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figure 10. Add permissions

c. Create a list of latitude and longitude coordinates, which will later be the determining point for the location to be paired with geofencing. In this example, a constant will be created from the HashMap named AREA_LANDMARKS. In the constant, values in the form of place names will be entered, and coordinate values in the form of latitude and longitude. An example can be seen in figure 11.

```java
public static final HashMap<String, LatLng> AREA_LANDMARKS = new HashMap<>();
static {
    AREA_LANDMARKS.put("Mall", new LatLng(-6.1773686, 106.7884836));
}
```

Figure 11. Location determinant point

d. Then a method is created to register the coordinate points that have been listed to be registered in geofencing. In this process, we will make settings when adding geofencing data, such as determining the latitude, longitude, and radius of the area to be registered, which are sourced from the constant AREA_LANDMARK. In addition, we need to set the active duration of the registered coordinate points and determine what types of events will function when the user enters the geofencing area. Suppose the event enters the area with the code Geofence. GEOFENCE_TRANSITION_ENTER. Then the action event that will be read by the geofencing system is when the user enters the geofencing area. The researcher created a method named populateGeofenceList(), as shown in figure 12.

```java
private void populateGeofenceList() {
    for (Map.Entry<String, LatLng> entry : MyConstant.AREA_LANDMARKS.entrySet()) {
        mGeofenceList.add(new Geofence.Builder()
                // Set the request ID of the geofence. This is a string to identify this
                // geofence.
                .setRequestId(entry.getKey())

                // Set the circular region of this geofence.
                .setCircularRegion(
                        entry.getValue().latitude,
                        entry.getValue().longitude,
                        MyConstant.GEOFENCE_RADIUS_IN_METERS
                )

                // Set the expiration duration of the geofence. This geofence gets automatically
                // removed after this period of time.
                //.setExpirationDuration(MyConstant.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
                .setExpirationDuration(Geofence.NEVER_EXPIRE)

                // Set the transition types of interest. Alerts are only generated for these
                // transition. We track entry and exit transitions in this sample.
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
                        Geofence.GEOFENCE_TRANSITION_EXIT)

                // Create the geofence.
                .build());

    }
}
```

Figure 12. Populate geofence list

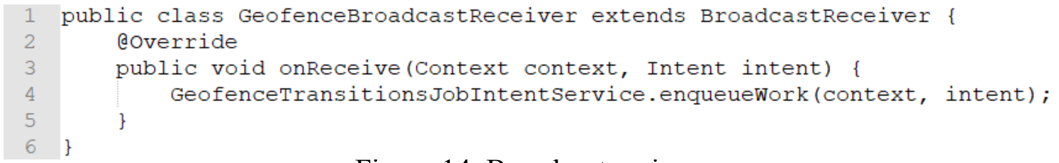
e. Add a geofencing area by using the GeofencingClient.addGeofences() method. This method gives the first parameter an object from the GeofencingRequest class, and the second parameter is an object from the PendingIntent class. An example of the program code can be seen in figure 13.

```java
mGeofencingClient.addGeofences(getGeofencingRequest(), getGeofencePendingIntent())
                .addOnCompleteListener(this);

```

Figure 13. Geofencing client

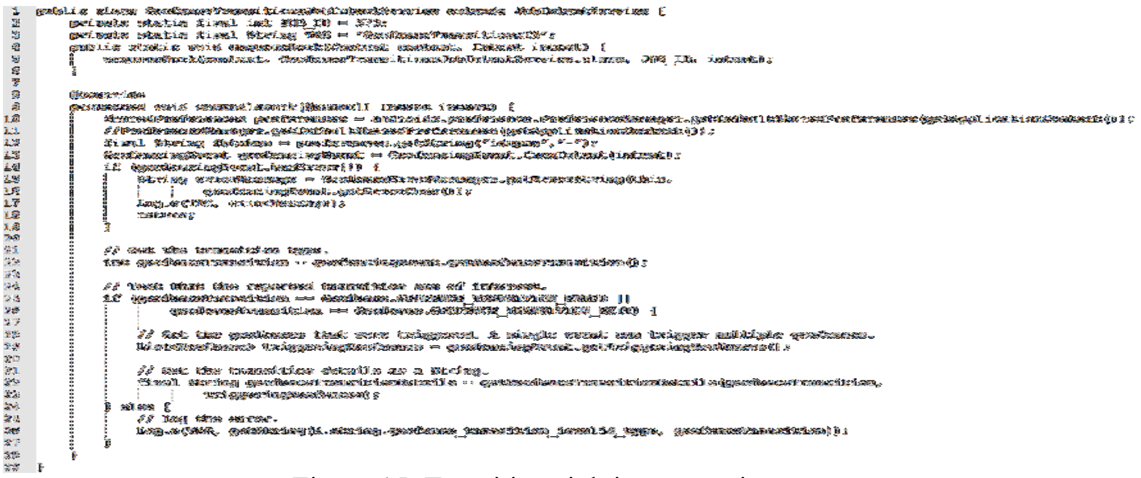
f. After registering the geofencing area, we will have to create a class to receive broadcast events. The class is named GeofenceBroadcastReceiver and is granted access to call functions on the BroadcastReceiver class. When the onReceive event is accessed with the context and intent parameters, it will run a geofencing transition event. The program code can be seen in figure 14.



```java
public class GeofenceBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        GeofenceTransitionsJobIntentService.enqueueWork(context, intent);
    }
}
```

Figure 14. Broadcast reciever

g. The next step is to create a class to handle geofencing transition events. This class will accept an intent and context and is named GeofenceTransitionsJobIntentService. That class will be added to run functions on the JobIntentService class. The function of this class is to find out the type of transition in geofencing that is in progress. In addition, this class can provide certain events on one of the transition types that meet predefined conditions. When there is a situation where the transition enters the geofencing area, certain notifications will be sent to the user. Some functions in the GeofenceTransitionsJobIntentService class can be seen in figure 15.



```java
public class GeofenceTransitionsJobIntentService extends JobIntentService {
    private static final int JOB_ID = 573;
    private static final String TAG = "GeofenceTransitionsIS";
    public static void enqueueWork(Context context, Intent intent) {
        enqueueWork(context, GeofenceTransitionsJobIntentService.class, JOB_ID, intent);
    }

    @Override
    protected void onHandleWork(@NonNull Intent intent) {
        SharedPreferences preferences = android.preference.PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        //PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        final String [illegible] = preferences.getString("[illegible]", "-");
        GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);
        if (geofencingEvent.hasError()) {
            String errorMessage = GeofenceErrorMessages.getErrorString(this,
                    geofencingEvent.getErrorCode());
            Log.e(TAG, errorMessage);
            return;
        }

        // Get the transition type.
        int geofenceTransition = geofencingEvent.getGeofenceTransition();

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger multiple geofences.
            List<Geofence> triggeringGeofences = geofencingEvent.getTriggeringGeofences();

            // Get the transition details as a String.
            final String geofenceTransitionDetails = getGeofenceTransitionDetails(this, geofenceTransition,
                    triggeringGeofences);
        } else {
            // Log the error.
            Log.e(TAG, getString(R.string.geofence_transition_invalid_type, geofenceTransition));
        }
    }
}
```

Figure 15. Transitions job intent service

4. CONCLUSION

Based on this research, we can conclude that after we apply geofencing technology by creating an Android-based mobile application, we can send information in the form of advertisements in a predetermined area, in this case, a shopping center area. Shopping center managers can be more effective and massive in sending advertisements to target consumers in a predetermined area. In this way, it is hoped that visitors to the shopping center will get the latest information and advertisements, and then they will be interested in shopping a lot in the area.

On the other hand, the majority of users consider geofencing technology effective in delivering promotional ads based on their location, with ad relevance being the main influencing factor. However, there is still a small percentage of users expressing concerns about privacy and interference with this geofencing technology.

5. SUGGESTED

In addition, the findings have implications for mobile app developers, dummies, and policymakers. Developers must ensure that geofencing technology is used in a transparent manner and protects user privacy. Advertisers should consider the relevance of their ads when using geofencing technology to target users based on their location. Policymakers should consider the need for regulations to protect user privacy in the context of geofencing technologies. The geofencing system uses location as a trigger for event determination. On the other hand, it states that user location information is private for application users, and Google is very concerned about privacy. Therefore, the need to provide a facility to confirm consent from application users when using this application can reduce the abuse of privacy. If the user does not allow the application to obtain location information, then this system cannot be used.

This application is made using Android OS; in subsequent development, it can use iOS or hybrid operating systems such as ReactNative. The API used in the development of this application is the Google API. For future development, you can use other APIs such as Amazon Location, Microsoft Azure Maps Spatial, and others.

REFERENCES

[1] P. Deshmukh, A. Bhajibhakre, S. Gambhire, A. Channe, and N. Deshpande, "Survey of Geofencing Algorithms," *Int. J. Comput. Sci. Eng. Tech.*, vol. 3, no. 2, 2018.

[2] P. M. Ekka, "A review of observation method in data collection process," *IJRTI Int. J. Res. Trends Innov.*, vol. 6, no. 12, 2021.

[3] E. A. R. Adhabi and C. B. L. Anozie, "Literature Review for the Type of Interview in Qualitative Research," *Int. J. Educ.*, vol. 9, no. 3, 2017, doi: 10.5296/ije.v9i3.11483.

[4] H. Snyder, "Literature review as a research methodology: An overview and guidelines," *J. Bus. Res.*, vol. 104, 2019, doi: 10.1016/j.jbusres.2019.07.039.

[5] I. Jovanovic, "Software Testing Methods and Techniques," *IPSI BgD Trans. Internet Res.*, vol. 5, no. 1, 2009.

[6] M. W. Sari, B. Santoso, and M. N. A. Azman, "Implementing Geo Positioning System for Children Tracking Location Monitoring based on Android," *Sci. J. Informatics*, vol. 8, no. 1, 2021, doi: 10.15294/sji.v8i1.27436.

[7] A. G. Buja, N. A. Deraman, K. A. F. Abu Samah, M. N. H. Hasrol Jono, M. A. M. Isa, and S. Bin Saad, "Enhancing a mobile application with contextual information delivery using proximity beacon: A preliminary study at a tourist destination," *Int. J. Adv. Technol. Eng. Explor.*, vol. 8, no. 78, 2021, doi: 10.19101/IJATEE.2020.762186.

[8] A. Permana, A. Hariyadi, and N. Zakaria, "Implementasi Teknologi Geofencing Untuk Pengawasan terhadap Lansia Menggunakan Sarung Lengan Berbasis Mikrokontroler dan Android," *J. Jartel J. Jar. Telekomun.*, vol. 10, no. 4, 2020, doi: 10.33795/jartel.v10i4.26.

[9] M. Makhtar, R. Rosly, S. A. Fadzli, S. N. W. Shamsuddin, and A. A. Jamal, "Implementation of mobile attendance application using geo-fence technique," *ARPN J. Eng. Appl. Sci.*, vol. 11, no. 5, 2016.

[10] S. R. Garzon, B. Deva, G. Pilz, and S. Medack, "Infrastructure-assisted geofencing: Proactive location-based services with thin mobile clients and smart servers," in

Proceedings - 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015, 2015. doi: 10.1109/MobileCloud.2015.31.