# BAB V

# KESIMPULAN DAN SARAN

## V.1. Kesimpulan

Dari hasil penelitian penulis, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Dengan di rancangnya sebuah aplikasi absensi menggunakan metode *Lock GPS* dengan *android* pada PT. Biro Klasifikasi Indonesia (persero) tidak perlu lagi mendatangi mesin absensi karena pegawai bisa langsung mengisi absensi melalu android pegawai.
2. Dengan adanya desain sistem, *PHP* serta *MySQL* dan sistem operasi *Android* dan di dungkung dengan *Lock GPS* pada aplikasi absensi perusahaan bisa mengetahui lokasi dan keberadaan pegawai. maka di rasa sangat efektif bagi pegawai PT. Biro Klasifikasi Indonesia karena tidak perlu lagi mendatangi mesin absensi dan efesien karena tidak membuang waktu dan menganggu waktu jam kerja pegawai.

## V.2. Saran

Dari kesimpulan yang ada, maka dapat dikemukakan saran-saran yang akan sangat membantu untuk pengembangan sistem ini selanjutnya:

1. Sebaiknya kedepanya tidak di perlukan lagi button refresh agar data absensi terupdate dengan sendirinya.
2. Sebaiknya nanti tidak perlu menunggu notifkasi agar pegawai langsung bisa mengisi absensi.
3. Sebaiknya  pada menu lokasi kerja memiliki button untuk pencari lokasi agar bisa langsung menampilkan pada titik lokasi yang di inginkan.

# DAFTAR PUSTAKA


Husain, A., Prastian, A. H. A., & Ramadhan, A. (2017). ***Perancangan sistem absensi online menggunakan android guna mempercepat proses kehadiran karyawan Pada PT. Sintech berkah abadi***. Technomedia Journal, 2(1), 105-116.

Khoir, S. A., Yudhana, A., & Sunardi, S. (2020). ***Implementasi GPS (Global Positioning System) Pada Presensi Berbasis Android DI BMT Insan Mandiri. J-SAKTI*** (Jurnal Sains Komputer dan Informatika), 4(1), 9-17.

Manu, G. A., & Benufinit, Y. A. (2020). ***Pengembangan Sistem Absensi Online Berbasis WEB Menggunakan Maps Javasripts Api.*** Jurnal Pendidikan Teknologi Informasi (JUKANTI), 3(2), 9-16

Mulyadi, E., Trihariprasetya, A., & Wiryawan, I. G. (2020). ***Penerapan Sistem Presensi Mobile Dengan Menggunakan Sensor GPS (Klinik Pratama X Di Jember).***Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI, 9(1), 11-20.

Sobarnas, M. A. (2020). ***Penerapan Geolokasi Pada Absensi Fasilitator Program Padat Karya Pemerintahan Yang Tersebar Di Seluruh Wilayah Indonesia.*** INFOTECH: Jurnal Informatika & Teknologi, 1(2), 116-126.

Sonny, S., & Rizki, S. N. (2021). ***Pengembangan Sistem Presensi Sistem Karya Dengan Teknologi GPS Berbasis WEB Pada PT BPR Dana Makmur Batam.***Computer and Science Industrial Engineering (COMASIE), 4(4), 52-58

## LISTING PROGRAM

MainActivity.java

```
package com.saliim.absensimobile;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;

import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.provider.MediaStore;
import android.provider.Settings;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingClient;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.snackbar.Snackbar;
import com.saliim.absensimobile.adapter.AbsensiPerUserAdapter;
```

```
import com.saliim.absensimobile.api.API;
import com.saliim.absensimobile.loginRegister.LoginActivity;
import com.saliim.absensimobile.model.absensi.DataAbsenPerUser;
import com.saliim.absensimobile.model.absensi.UpdateAbsen;
import com.saliim.absensimobile.model.lokasi.DataLokasi;
import com.saliim.absensimobile.model.uploadGambar.BaseResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import okhttp3.ResponseBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

import static com.saliim.absensimobile.loginRegister.LoginActivity.MY_LOGIN_PREF;
import static
com.saliim.absensimobile.loginRegister.LoginActivity.MY_LOGIN_PREF_KEY;


public class MainActivity extends AppCompatActivity implements
OnCompleteListener<Void> {

    private static final String TAG = MainActivity.class.getSimpleName();

    private static final int REQUEST_PERMISSIONS_REQUEST_CODE = 34;

    /**
     * Tracks whether the user requested to add or remove geofences, or to do neither.
     */
    private enum PendingGeofenceTask {
        ADD, NONE
    }

    /**
     * Provides access to the Geofencing API.
     */
    private GeofencingClient mGeofencingClient;

    /**
     * The list of geofences used in this sample.
```

```
     */
    private ArrayList<Geofence> mGeofenceList;

    /**
     * Used when requesting to add or remove geofences.
     */
    private PendingIntent mGeofencePendingIntent;

    // Buttons for kicking off the process of adding or removing geofences.
//    private Button mAddGeofencesButton;
//    private Button mRemoveGeofencesButton;

    private PendingGeofenceTask mPendingGeofenceTask = PendingGeofenceTask.NONE;

    private Toolbar toolbar;

    private static final int REQUEST_IMAGE_CAPTURE = 1;
    private static final int REQUEST_STORAGE_PERMISSION = 1;

    private static final String TYPE_1 = "multipart";
    private static final String TYPE_2 = "base64";

    private static final String FILE_PROVIDER_AUTHORITY =
"com.google.android.provider";

    String mTempPhotoPath;
    String savedImagePath;

    public static boolean btnAvailable = false;

    private Bitmap mResultsBitmap;

    int absenCounter = 0;

//    private Uri uri;

    String dates, dateNow, idAbsen;
    Date date;

    ImageView imageView, mPhoto, logout;
    Button mAbsen, mClear;
    RecyclerView recylerAbsen;
    View line;

    ArrayList dataAbsensiperUsers;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
```

```
        toolbar = findViewById(R.id.toolbars);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Absen "+ LoginActivity.name);

        mPhoto = findViewById(R.id.btn_tp);
        imageView = findViewById(R.id.imageView);
        mAbsen = findViewById(R.id.btn_absen);
        mClear = findViewById(R.id.clear);
        recylerAbsen = findViewById(R.id.recycler_absensi_perUser);
        line = findViewById(R.id.line1);
        logout = findViewById(R.id.btn_logout_user);

        imageView.setVisibility(View.GONE);
        mAbsen.setVisibility(View.GONE);
        mClear.setVisibility(View.GONE);
        mPhoto.setVisibility(View.VISIBLE);
        logout.setVisibility(View.VISIBLE);
        recylerAbsen.setVisibility(View.VISIBLE);
        line.setVisibility(View.VISIBLE);

        getDataAbsenPerUser();
        dateNow = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault()).format(new
Date());

        Log.i("hari ini", dateNow);


        // Empty list for storing geofences.
        mGeofenceList = new ArrayList<>();

        // Initially set the PendingIntent used in addGeofences() and removeGeofences() to null.
        mGeofencePendingIntent = null;

        // Get the geofences used. Geofence data is hard coded in this sample.
        mGeofencingClient = LocationServices.getGeofencingClient(this);

        mAbsen.setOnClickListener(v ->{

            if (mResultsBitmap != null){
                String encode = ImageUtils.bitmapToBase64String(mResultsBitmap, 100);
                uploadBase64(encode);
            }else{
                Toast.makeText(this, "tidak ada gambar", Toast.LENGTH_SHORT).show();
            }

            Log.d("mResultsBitmap", String.valueOf(mResultsBitmap));
            });

        mClear.setOnClickListener(v -> clear());
```

```
        logout.setOnClickListener(v -> {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder .setTitle("Peringatan")
                    .setMessage("Anda Yakin ingin Logout?")
                    .setPositiveButton("Iya", (dialog, which) -> {

                        SharedPreferences preferences = getSharedPreferences(MY_LOGIN_PREF,
Context.MODE_PRIVATE);
                        preferences.edit().remove(MY_LOGIN_PREF_KEY).apply();

                        startActivity(new Intent(MainActivity.this, LoginActivity.class));
                        finish();
                    })
                    .setNegativeButton("Tidak", (dialog, which) -> {
                        dialog.dismiss();
                    });

            builder.create();
            builder.show();
        });

    }

    @Override
    public void onStart() {
        super.onStart();

        API.dataLokasi().enqueue(new Callback<ArrayList<DataLokasi>>() {
            @Override
            public void onResponse(Call<ArrayList<DataLokasi>> call,
Response<ArrayList<DataLokasi>> response) {
                if (response.code() == 200){
                    Constants.dataLokasis = response.body();

                    populateGeofenceList();

                    Geofenceattach();
                }
            }

            @Override
            public void onFailure(Call<ArrayList<DataLokasi>> call, Throwable t) {

            }
        });

        if (!checkPermissions()) {
            requestPermissions();
        } else {
```

```
            performPendingGeofenceTask();
        }
    }

    public void Absens(View view){

        if (!btnAvailable){
            Toast.makeText(this, "Anda masih di luar lokasi", Toast.LENGTH_SHORT).show();
//            Log.i("Geo sekarang", GeofenceTransitionsJobIntentService.namaLokasi);
            return;
        }

//        Log.i("hari itu", String.valueOf(date));
//        Log.i("hari itu2", dates);
        if (dateNow.equals(dates)){
            updateAbsen();
        }else{
            String gambarNull = "-";

            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder .setMessage("Ingin Menambahkan Foto pada Absen?")
                    .setPositiveButton("Iya", (dialog, which) -> {
                        // Check for the external storage permission
                        if (ContextCompat.checkSelfPermission(getApplicationContext(),
                                Manifest.permission.WRITE_EXTERNAL_STORAGE)
                                != PackageManager.PERMISSION_GRANTED) {

                            // If you do not have permission, request it
                            ActivityCompat.requestPermissions(this,
                                    new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.CAMERA},
                                    REQUEST_STORAGE_PERMISSION);
                        }else{
                            launchCamera();
                        }
                    })
                    .setNegativeButton("Tidak", (dialog, which) -> {
                        dialog.dismiss();
                        IsiAbsen(gambarNull);
                    });
            builder.create();
            builder.show();
        }

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        // If the image capture activity was called and was successful
```

```
        if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==
Activity.RESULT_OK) {

            processAndSetImage();

            imageView.setImageBitmap(mResultsBitmap);
            saveImage(this, mResultsBitmap);

        }
    }

    /**
     * Builds and returns a GeofencingRequest. Specifies the list of geofences to be monitored.
     * Also specifies how the geofence notifications are initially triggered.
     */
    private GeofencingRequest getGeofencingRequest() {
        GeofencingRequest.Builder builder = new GeofencingRequest.Builder();

        // The INITIAL_TRIGGER_ENTER flag indicates that geofencing service should
trigger a
        // GEOFENCE_TRANSITION_ENTER notification when the geofence is added and if
the device
        // is already inside that geofence.
        builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);

        // Add the geofences to be monitored by geofencing service.
        builder.addGeofences(mGeofenceList);

        // Return a GeofencingRequest.
        return builder.build();
    }

    public void IsiAbsen(String gambarPath){

        String id_user = LoginActivity.id;
        String nama = LoginActivity.name;
        String lokasi = GeofenceTransitionsJobIntentService.namaLokasi;
        String status = "HADIR";
        String gambar = gambarPath;

        API.addAbsen(nama, lokasi, status, id_user, gambar).enqueue(new
Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody>
response) {
                if (response.code() == 200){
                    Log.i("absensi", "" + response.body());
                    Toast.makeText(MainActivity.this, "Terima Kasih",
Toast.LENGTH_SHORT).show();
                    getDataAbsenPerUser();
```

```
                absenCounter++;
                clear();
            } else {
                Toast.makeText(MainActivity.this, "Silahkan Coba Lagi",
Toast.LENGTH_SHORT).show();
                clear();
            }
        }

        @Override
        public void onFailure(Call<ResponseBody> call, Throwable t) {
            Toast.makeText(MainActivity.this, "Absen Gagal",
Toast.LENGTH_SHORT).show();
            clear();
        }
    });
  }

  /**
   * Adds geofences, which sets alerts to be notified when the device enters or exits one of
the
   * specified geofences. Handles the success or failure results returned by addGeofences().
   */
  public void Geofenceattach() {
    if (!checkPermissions()) {
      mPendingGeofenceTask = PendingGeofenceTask.ADD;
      requestPermissions();
      return;
    }
    addGeofences();
  }

  /**
   * Adds geofences. This method should be called after the user has granted the location
   * permission.
   */
  @SuppressWarnings("MissingPermission")
  private void addGeofences() {
    if (!checkPermissions()) {
      showSnackbar(getString(R.string.insufficient_permissions));
      return;
    }

    mGeofencingClient.addGeofences(getGeofencingRequest(),
getGeofencePendingIntent())
            .addOnCompleteListener(this);
  }


  /**
```

```
     * Runs when the result of calling {@link #addGeofences()} and/or {@link
#/removeGeofences()}
     * is available.
     * @param task the resulting Task, containing either a result or error.
     */
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        mPendingGeofenceTask = PendingGeofenceTask.NONE;
        if (task.isSuccessful()) {
            updateGeofencesAdded(!getGeofencesAdded());

        } else {
            // Get the status code for the error and log it using a user-friendly message.
            String errorMessage = GeofenceErrorMessages.getErrorString(this,
task.getException());
            Log.w(TAG, errorMessage);
        }
    }

    /**
     * Gets a PendingIntent to send with the request to add or remove Geofences. Location
Services
     * issues the Intent inside this PendingIntent whenever a geofence transition occurs for the
     * current list of geofences.
     *
     * @return A PendingIntent for the IntentService that handles geofence transitions.
     */
    private PendingIntent getGeofencePendingIntent() {
        // Reuse the PendingIntent if we already have it.
        if (mGeofencePendingIntent != null) {
            return mGeofencePendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back
when calling
        // addGeofences() and removeGeofences().
        mGeofencePendingIntent = PendingIntent.getBroadcast(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
        return mGeofencePendingIntent;
    }

    /**
     * This sample hard codes geofence data. A real app might dynamically create geofences
based on
     * the user's location.
     */

//    ArrayList<DataLokasi> datalokasi = new ArrayList<>();

    private void populateGeofenceList() {
```

```
        for (DataLokasi dataLokasi: Constants.dataLokasis) {

            Log.d("aaa", dataLokasi.toString());

            mGeofenceList.add(new Geofence.Builder()
                    // Set the request ID of the geofence. This is a string to identify this
                    // geofence.
                    .setRequestId(dataLokasi.getLokasi())

                    // Set the circular region of this geofence.
                    .setCircularRegion(
                            Double.parseDouble(dataLokasi.getLatitude()),
                            Double.parseDouble(dataLokasi.getLongitude()),
                            Float.parseFloat(dataLokasi.getRadius())
                    )

                    // Set the expiration duration of the geofence. This geofence gets automatically
                    // removed after this period of time.

.setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)

                    // 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());
        }
        Log.d("GeofenceList", String.valueOf(mGeofenceList));
    }

    /**
     * Shows a {@link Snackbar} using {@code text}.
     *
     * @param text The Snackbar text.
     */
    private void showSnackbar(final String text) {
        View container = findViewById(android.R.id.content);
        if (container != null) {
            Snackbar.make(container, text, Snackbar.LENGTH_LONG).show();
        }
    }

    /**
     * Shows a {@link Snackbar}.
     *
     * @param mainTextStringId The id for the string resource for the Snackbar text.
     * @param actionStringId   The text of the action item.
     * @param listener         The listener associated with the Snackbar action.
```

```
 */
private void showSnackbar(final int mainTextStringId, final int actionStringId,
                          View.OnClickListener listener) {
    Snackbar.make(
            findViewById(android.R.id.content),
            getString(mainTextStringId),
            Snackbar.LENGTH_INDEFINITE)
            .setAction(getString(actionStringId), listener).show();
}

/**
 * Returns true if geofences were added, otherwise false.
 */
private boolean getGeofencesAdded() {
    return PreferenceManager.getDefaultSharedPreferences(this).getBoolean(
            Constants.GEOFENCES_ADDED_KEY, false);
}

/**
 *
 *
 * @param added Whether geofences were added or removed.
 */
private void updateGeofencesAdded(boolean added) {
    PreferenceManager.getDefaultSharedPreferences(this)
            .edit()
            .putBoolean(Constants.GEOFENCES_ADDED_KEY, added)
            .apply();
}

/**
 * Performs the geofencing task that was pending until location permission was granted.
 */
private void performPendingGeofenceTask() {
    if (mPendingGeofenceTask == PendingGeofenceTask.ADD) {
        addGeofences();
    } /*else if (mPendingGeofenceTask == PendingGeofenceTask.REMOVE) {
        removeGeofences();
    }*/
}

/**
 * Return the current state of the permissions needed.
 */
private boolean checkPermissions() {
    int permissionState = ActivityCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION);
    return permissionState == PackageManager.PERMISSION_GRANTED;
}
```

```
  private void requestPermissions() {
    boolean shouldProvideRationale =
        ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.ACCESS_FINE_LOCATION);

    // Provide an additional rationale to the user. This would happen if the user denied the
    // request previously, but didn't check the "Don't ask again" checkbox.
    if (shouldProvideRationale) {
      Log.i(TAG, "Displaying permission rationale to provide additional context.");
      showSnackbar(R.string.permission_rationale, android.R.string.ok,
          new View.OnClickListener() {
            @Override
            public void onClick(View view) {
              // Request permission
              ActivityCompat.requestPermissions(MainActivity.this,
                  new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                  REQUEST_PERMISSIONS_REQUEST_CODE);
            }
          });
    } else {
      Log.i(TAG, "Requesting permission");
      // Request permission. It's possible this can be auto answered if device policy
      // sets the permission in a given state or the user denied the permission
      // previously and checked "Never ask again".
      ActivityCompat.requestPermissions(MainActivity.this,
          new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
          REQUEST_PERMISSIONS_REQUEST_CODE);
    }
  }

  /**
   * Callback received when a permissions request has been completed.
   */
  @Override
  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                         @NonNull int[] grantResults) {
    Log.i(TAG, "onRequestPermissionResult");
    switch (requestCode){
      case REQUEST_STORAGE_PERMISSION:{
        if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
          // If you get permission, launch the camera
          launchCamera();
        } else {
          // If you do not get permission, show a Toast
          Toast.makeText(this, R.string.permission_denied,
Toast.LENGTH_SHORT).show();
        }
        break;
      }
```

```
            case REQUEST_PERMISSIONS_REQUEST_CODE:{
                if (grantResults.length <= 0) {
                    // If user interaction was interrupted, the permission request is cancelled and you
                    // receive empty arrays.
                    Log.i(TAG, "User interaction was cancelled.");
                } else if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    Log.i(TAG, "Permission granted.");
                    performPendingGeofenceTask();
                } else {
                    // Permission denied.

                    // Notify the user via a SnackBar that they have rejected a core permission for the
                    // app, which makes the Activity useless. In a real app, core permissions would
                    // typically be best requested during a welcome-screen flow.

                    // Additionally, it is important to remember that a permission might have been
                    // rejected without asking the user for permission (device policy or "Never ask
                    // again" prompts). Therefore, a user interface affordance is typically
implemented
                    // when permissions are denied. Otherwise, your app could appear unresponsive
to
                    // touches or interactions which have required permissions.
                    showSnackbar(R.string.permission_denied_explanation, R.string.settings,
                            new View.OnClickListener() {
                                @Override
                                public void onClick(View view) {
                                    // Build intent that displays the App settings screen.
                                    Intent intent = new Intent();
                                    intent.setAction(
                                            Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                                    Uri uri = Uri.fromParts("package",
                                            BuildConfig.APPLICATION_ID, null);
                                    intent.setData(uri);
                                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                    startActivity(intent);
                                }
                            });
                    mPendingGeofenceTask = PendingGeofenceTask.NONE;
                }
                break;
            }
        }
    }

    //take picture and save image part
    private void launchCamera(){
        // Create the capture image intent
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

//        uri = Uri.fromFile();
```

```
        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            // Create the temporary File where the photo should go
            File photoFile = null;
            try {
                photoFile = createImageFile(this);
            } catch (IOException ex) {
                // Error occurred while creating the File
                ex.printStackTrace();
            }
            // Continue only if the File was successfully created
            if (photoFile != null) {

                // Get the path of the temporary file
                mTempPhotoPath = photoFile.getAbsolutePath();

                // Get the content URI for the image file
                Uri photoURI = FileProvider.getUriForFile(this,
                        FILE_PROVIDER_AUTHORITY,
                        photoFile);

                // Add the URI so the camera can store the image
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);

                // Launch the camera activity
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
            }
        }
    }


    String saveImage(Context context, Bitmap image){
        savedImagePath = null;

        // Create the new file in the external storage
        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss",
                Locale.getDefault()).format(new Date());
        String imageFileName = "JPEG_" + timeStamp + ".jpg";
        File storageDir = new File(

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
                + "/MyCamera");
        boolean success = true;
        if (!storageDir.exists()){
            success = storageDir.mkdirs();
        }

        if (success) {
            File imageFile = new File(storageDir, imageFileName);
```

```
            savedImagePath = imageFile.getAbsolutePath();
            try {
                OutputStream fOut = new FileOutputStream(imageFile);
                image.compress(Bitmap.CompressFormat.JPEG, 100, fOut);
                fOut.close();
            } catch (Exception e) {
                e.printStackTrace();
            }

            // Add the image to the system gallery
            galleryAddPic(context, savedImagePath);

            // Show a Toast with the save location
            // String savedMessage = context.getString(R.string.saved_message,
savedImagePath);

        }
        Log.d("image", savedImagePath);
        return savedImagePath;
    }

    private void galleryAddPic(Context context, String imagePath) {
        Intent mediaScanIntent = new
Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
        File f = new File(imagePath);
        Uri contentUri = Uri.fromFile(f);
        mediaScanIntent.setData(contentUri);
        context.sendBroadcast(mediaScanIntent);
    }

    public void clear(){
        imageView.setImageResource(0);
        imageView.setVisibility(View.GONE);
        mAbsen.setVisibility(View.GONE);
        mClear.setVisibility(View.GONE);
        mPhoto.setVisibility(View.VISIBLE);
        logout.setVisibility(View.VISIBLE);
        recylerAbsen.setVisibility(View.VISIBLE);
        line.setVisibility(View.VISIBLE);

    }

    private void processAndSetImage() {

        // Toggle Visibility of the views
        mPhoto.setVisibility(View.GONE);
        logout.setVisibility(View.GONE);
        recylerAbsen.setVisibility(View.GONE);
        line.setVisibility(View.GONE);
        mAbsen.setVisibility(View.VISIBLE);
```

```
        mClear.setVisibility(View.VISIBLE);
        imageView.setBackgroundResource(R.drawable.ic_picture);
        imageView.setVisibility(View.VISIBLE);

        // Resample the saved image to fit the ImageView
        mResultsBitmap = resamplePic(this, mTempPhotoPath);

    }

    private File createImageFile(Context context) throws IOException {
        // Create an image file name
        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss",
Locale.getDefault()).format(new Date());
        String imageFileName = "JPEG_" + timeStamp + "_";
        File storageDir = context.getExternalCacheDir();

        return File.createTempFile(
                imageFileName,  /* prefix */
                ".jpg",         /* suffix */
                storageDir      /* directory */
        );

    }

    Bitmap resamplePic(Context context, String imagePath) {

        // Get device screen size information
        DisplayMetrics metrics = new DisplayMetrics();
        WindowManager manager = (WindowManager)
context.getSystemService(Context.WINDOW_SERVICE);
        manager.getDefaultDisplay().getMetrics(metrics);

        int targetH = metrics.heightPixels;
        int targetW = metrics.widthPixels;

        // Get the dimensions of the original bitmap
        BitmapFactory.Options bmOptions = new BitmapFactory.Options();
        bmOptions.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(imagePath, bmOptions);
        int photoW = bmOptions.outWidth;
        int photoH = bmOptions.outHeight;

        // Determine how much to scale down the image
        int scaleFactor = Math.min(photoW / targetW, photoH / targetH);

        // Decode the image file into a Bitmap sized to fill the View
        bmOptions.inJustDecodeBounds = false;
        bmOptions.inSampleSize = scaleFactor;

        return BitmapFactory.decodeFile(imagePath);
```

```
    }

    //Upload gambar dgn Base64
    private void uploadBase64(String imgBase64) {
        ProgressDialog progressDialog = new ProgressDialog(MainActivity.this);
        progressDialog.setMessage("Memuat Data Absen.....");
        progressDialog.show();

        API.uploadPhotoBase64(TYPE_2, imgBase64).enqueue(new
Callback<BaseResponse>() {
            @Override
            public void onResponse(Call<BaseResponse> call, Response<BaseResponse>
response) {
                BaseResponse baseResponse = (BaseResponse) response.body();

                if(baseResponse != null) {
                    String gambarPath = baseResponse.getPath();
                    IsiAbsen(gambarPath);
                    progressDialog.dismiss();
                    Log.d("baseResponse", baseResponse.getPath());
                }
            }

            @Override
            public void onFailure(Call<BaseResponse> call, Throwable t) {
                progressDialog.dismiss();
                t.printStackTrace();
            }
        });
    }

    private void getDataAbsenPerUser() {
        String id_user = LoginActivity.id;

        API.dataAbsenPerUser(id_user).enqueue(new
Callback<ArrayList<DataAbsenPerUser>>() {
            @Override
            public void onResponse(Call<ArrayList<DataAbsenPerUser>> call,
Response<ArrayList<DataAbsenPerUser>> response) {
                if(response.code() == 200){
                    dataAbsensiperUsers = response.body();
                    Log.d("dataAbsensiperUser", String.valueOf(dataAbsensiperUsers));

                    if (dataAbsensiperUsers.isEmpty()){
                        Toast.makeText(MainActivity.this, "data kosong",
Toast.LENGTH_SHORT).show();
                    }else{
                        recylerAbsen.hasFixedSize();
                        recylerAbsen.setLayoutManager(new
LinearLayoutManager(MainActivity.this));
```

```
                recylerAbsen.setAdapter(new AbsensiPerUserAdapter(dataAbsensiperUsers));

                dates = response.body().get(0).getJamMasuk();
                idAbsen = response.body().get(0).getIdAbsen();

                try {
                    date = new SimpleDateFormat("yyyy-MM-dd").parse(dates);
                } catch (ParseException e) {
                    e.printStackTrace();
                }

                dates = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(date);

            }

        }
    }

    @Override
    public void onFailure(Call<ArrayList<DataAbsenPerUser>> call, Throwable t) {

    }
  });
 }

 private void updateAbsen() {
    ProgressDialog progressDialog = new ProgressDialog(MainActivity.this);
    progressDialog.setMessage("memperbarui jam pulang .....");
    progressDialog.show();

    API.updateAbsen(idAbsen).enqueue(new Callback<UpdateAbsen>() {
      @Override
      public void onResponse(Call<UpdateAbsen> call, Response<UpdateAbsen>
response) {
        if (response.code() == 200){
          Log.i("update absen", String.valueOf(response.body()));
          getDataAbsenPerUser();
          Toast.makeText(MainActivity.this, "Jam Pulang Telah Di Update",
Toast.LENGTH_SHORT).show();
          progressDialog.dismiss();
        }else{
          Toast.makeText(MainActivity.this, "Jam Pulang Gagal DI Update",
Toast.LENGTH_SHORT).show();
          progressDialog.dismiss();
        }
      }

      @Override
      public void onFailure(Call<UpdateAbsen> call, Throwable t) {
```

```
            Toast.makeText(MainActivity.this, "gagal Update",
Toast.LENGTH_SHORT).show();
            progressDialog.dismiss();
        }
      });
   }

}
```

GeofenceTransitionsJobIntentService.java

```
package com.saliim.absensimobile;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Build;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;

import androidx.core.app.JobIntentService;
import androidx.core.app.NotificationCompat;
import androidx.core.app.TaskStackBuilder;

import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingEvent;

import java.util.ArrayList;
import java.util.List;

public class GeofenceTransitionsJobIntentService extends JobIntentService {
   private static final int JOB_ID = 573;

   private static final String TAG = "GeofenceTransitionsIS";

   private static final String CHANNEL_ID = "channel_01";

   public static String namaLokasi;

   /**
    * Convenience method for enqueuing work in to this service.
    */
   public static void enqueueWork(Context context, Intent intent) {
      enqueueWork(context, GeofenceTransitionsJobIntentService.class, JOB_ID, intent);
```

```
    }

    /**
     * Handles incoming intents.
     * @param intent sent by Location Services. This Intent is provided to Location
     *               Services (inside a PendingIntent) when addGeofences() is called.
     */
    @Override
    protected void onHandleWork(Intent intent) {
        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.
            String geofenceTransitionDetails = getGeofenceTransitionDetails(geofenceTransition,
                    triggeringGeofences);


            MainActivity.btnAvailable = true;
            // Send notification and log the transition details.
            sendNotification(geofenceTransitionDetails);
//            MainActivity.mPhoto.setEnabled(false);
            Log.i(TAG, geofenceTransitionDetails);
        } else {
            // Log the error.
            Log.e(TAG, getString(R.string.geofence_transition_invalid_type,
geofenceTransition));
        }
    }

    /**
     * Gets transition details and returns them as a formatted string.
     *
     * @param geofenceTransition    The ID of the geofence transition.
     * @param triggeringGeofences   The geofence(s) triggered.
```

```
     * @return                The transition details formatted as String.
     */
    private String getGeofenceTransitionDetails(
            int geofenceTransition,
            List<Geofence> triggeringGeofences) {

        String geofenceTransitionString = getTransitionString(geofenceTransition);

        // Get the Ids of each geofence that was triggered.
        ArrayList<String> triggeringGeofencesIdsList = new ArrayList<>();
        for (Geofence geofence : triggeringGeofences) {
            triggeringGeofencesIdsList.add(geofence.getRequestId());
            namaLokasi = geofence.getRequestId();
        }
        String triggeringGeofencesIdsString = TextUtils.join(", ",  triggeringGeofencesIdsList);

        return geofenceTransitionString + ": " + triggeringGeofencesIdsString;
    }

    /**
     * Posts a notification in the notification bar when a transition is detected.
     * If the user clicks the notification, control goes to the MainActivity.
     */
    private void sendNotification(String notificationDetails) {
        // Get an instance of the Notification manager
        NotificationManager mNotificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        // Android O requires a Notification Channel.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            CharSequence name = getString(R.string.app_name);
            // Create the channel for the notification
            NotificationChannel mChannel =
                    new NotificationChannel(CHANNEL_ID, name,
NotificationManager.IMPORTANCE_DEFAULT);

            // Set the Notification Channel for the Notification Manager.
            mNotificationManager.createNotificationChannel(mChannel);
        }

        // Create an explicit content Intent that starts the main Activity.
        Intent notificationIntent = new Intent(getApplicationContext(), MainActivity.class);

        // Construct a task stack.
        TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);

        // Add the main Activity to the task stack as the parent.
        stackBuilder.addParentStack(MainActivity.class);

        // Push the content Intent onto the stack.
```

```
        stackBuilder.addNextIntent(notificationIntent);

        // Get a PendingIntent containing the entire back stack.
        PendingIntent notificationPendingIntent =
                stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);

        // Get a notification builder that's compatible with platform versions >= 4
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this);

        // Define the notification settings.
        builder.setSmallIcon(R.drawable.ic_launcher_background)
                // In a real app, you may want to use a library like Volley
                // to decode the Bitmap.
                .setLargeIcon(BitmapFactory.decodeResource(getResources(),
                        R.drawable.ic_launcher_background))
                .setColor(Color.RED)
                .setContentTitle(notificationDetails)
                .setContentText(getString(R.string.geofence_transition_notification_text))
                .setContentIntent(notificationPendingIntent);

        // Set the Channel ID for Android O.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            builder.setChannelId(CHANNEL_ID); // Channel ID
        }

        // Dismiss notification once the user touches it.
        builder.setAutoCancel(true);

        // Issue the notification
        mNotificationManager.notify(0, builder.build());
    }

    /**
     * Maps geofence transition types to their human-readable equivalents.
     *
     * @param transitionType    A transition type constant defined in Geofence
     * @return                  A String indicating the type of transition
     */
    private String getTransitionString(int transitionType) {
        switch (transitionType) {
            case Geofence.GEOFENCE_TRANSITION_ENTER:
                return getString(R.string.geofence_transition_entered);
            case Geofence.GEOFENCE_TRANSITION_EXIT:
                return getString(R.string.geofence_transition_exited);
            default:
                return getString(R.string.unknown_geofence_transition);
        }
    }
}
```