LAMPIRAN

A. Fitur Geofencing & Haversine Formula

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:geocoding/geocoding.dart';
import 'package:geolocator/geolocator.dart';
import 'package:get/get.dart';
import 'package:intl/intl.dart';
import 'package:presence/app/widgets/dialog/custom_alert_dialog.dart';
import 'package:presence/app/widgets/toast/custom_toast.dart';
import 'package:presence/company_data.dart';

class PresenceController extends GetxController {
  RxBool isLoading = false.obs;

  FirebaseAuth auth = FirebaseAuth.instance;
  FirebaseFirestore firestore = FirebaseFirestore.instance;

  presence() async {
    isLoading.value = true;
    Map<String, dynamic> determinePosition = await _determinePosition();
    if (!determinePosition["error"]) {
      Position position = determinePosition["position"];
      List<Placemark> placemarks =
          await placemarkFromCoordinates(position.latitude,
position.longitude);
      String address =
          "${placemarks.first.street}, ${placemarks.first.subLocality},
${placemarks.first.locality}";
      double distance = Geolocator.distanceBetween(
          CompanyData.office['latitude'],
          CompanyData.office['longitude'],
          position.latitude,
          position.longitude);

      await updatePosition(position, address);

      if (distance <= 200) {
        await processPresence(position, address, distance);
        isLoading.value = false;
      } else {
```

```
      isLoading.value = false;
      CustomToast.errorToast(
          'Tidak Dalam Area Presensi!', 'Minimal Jarak 200 M dari Sekolah');
    }
  } else {
    isLoading.value = false;
    CustomToast.errorToast("Error", determinePosition["message"]);
  }
}

firstPresence(
  CollectionReference<Map<String, dynamic>> presenceCollection,
  String todayDocId,
  Position position,
  String address,
  double distance,
  bool inArea,
) async {
  CustomAlertDialog.showPresenceAlert(
    title: "Are you want to check in?",
    message: "you need to confirm before you\ncan do presence now",
    onCancel: () => Get.back(),
    onConfirm: () async {
      await presenceCollection.doc(todayDocId).set(
        {
          "date": DateTime.now().toIso8601String(),
          "masuk": {
            "date": DateTime.now().toIso8601String(),
            "latitude": position.latitude,
            "longitude": position.longitude,
            "address": address,
            "in_area": inArea,
            "distance": distance,
          }
        },
      );
      Get.back();
      CustomToast.successToast("Success", "success check in");
    },
  );
}

checkinPresence(
```

```
  CollectionReference<Map<String, dynamic>> presenceCollection,
  String todayDocId,
  Position position,
  String address,
  double distance,
  bool inArea,
) async {
  CustomAlertDialog.showPresenceAlert(
    title: "Are you want to check in?",
    message: "you need to confirm before you\ncan do presence now",
    onCancel: () => Get.back(),
    onConfirm: () async {
      await presenceCollection.doc(todayDocId).set(
        {
          "date": DateTime.now().toIso8601String(),
          "masuk": {
            "date": DateTime.now().toIso8601String(),
            "latitude": position.latitude,
            "longitude": position.longitude,
            "address": address,
            "in_area": inArea,
            "distance": distance,
          }
        },
      );
      Get.back();
      CustomToast.successToast("Success", "success check in");
    },
  );
}

checkoutPresence(
  CollectionReference<Map<String, dynamic>> presenceCollection,
  String todayDocId,
  Position position,
  String address,
  double distance,
  bool inArea,
) async {
  CustomAlertDialog.showPresenceAlert(
    title: "Are you want to check out?",
    message: "you need to confirm before you\ncan do presence now",
    onCancel: () => Get.back(),
```

```
    onConfirm: () async {
      await presenceCollection.doc(todayDocId).update(
        {
          "keluar": {
            "date": DateTime.now().toIso8601String(),
            "latitude": position.latitude,
            "longitude": position.longitude,
            "address": address,
            "in_area": inArea,
            "distance": distance,
          }
        },
      );
      Get.back();
      CustomToast.successToast("Success", "success check out");
    },
  );
}

Future<void> processPresence(
    Position position, String address, double distance) async {
  String uid = auth.currentUser!.uid;
  String todayDocId =
      DateFormat.yMd().format(DateTime.now()).replaceAll("/", "-");

  CollectionReference<Map<String, dynamic>> presenceCollection =
      firestore.collection("pelatih").doc(uid).collection("presence");
  QuerySnapshot<Map<String, dynamic>> snapshotPreference =
      await presenceCollection.get();

  bool inArea = false;
  if (distance <= 200) {
    inArea = true;
  }

  if (snapshotPreference.docs.isEmpty) {

    firstPresence(
        presenceCollection, todayDocId, position, address, distance, inArea);
  } else {
    DocumentSnapshot<Map<String, dynamic>> todayDoc =
        await presenceCollection.doc(todayDocId).get();
```

```
      if (todayDoc.exists == true) {
        Map<String, dynamic>? dataPresenceToday = todayDoc.data();

        if (dataPresenceToday?["keluar"] != null) {
          CustomToast.successToast(
              "Success", "you already check in and check out");
        } else {
          // case : already check in and not yet check out ( check out )
          checkoutPresence(presenceCollection, todayDocId, position,
address,
              distance, inArea);
        }
      } else {
        checkinPresence(presenceCollection, todayDocId, position, address,
            distance, inArea);
      }
    }
  }

  Future<void> updatePosition(Position position, String address) async {
    String uid = auth.currentUser!.uid;
    await firestore.collection("pelatih").doc(uid).update({
      "position": {
        "latitude": position.latitude,
        "longitude": position.longitude,
      },
      "address": address,
    });
  }

  Future<Map<String, dynamic>> _determinePosition() async {
    bool serviceEnabled;
    LocationPermission permission;

    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      return Future.error('Location services are disabled.');
    }

    permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
      permission = await Geolocator.requestPermission();
      if (permission == LocationPermission.denied) {
```

```
      return {
        "message":
            "Tidak dapat mengakses karena anda menolak permintaan lokasi",
        "error": true,
      };
    }
  }

  if (permission == LocationPermission.deniedForever) {
    return {
      "message":
          "Location permissions are permanently denied, we cannot request
permissions.",
      "error": true,
    };
  }

  Position position = await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.bestForNavigation);
  return {
    "position": position,
    "message": "Berhasil mendapatkan posisi device",
    "error": false,
  };
 }
}
```