```
package com.example.careunityportal;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.app.NotificationCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.DialogFragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.api.Scope;
import com.google.android.gms.fitness.Fitness;
import com.google.android.gms.fitness.FitnessOptions;
import com.google.android.gms.fitness.data.DataPoint;
import com.google.android.gms.fitness.data.DataSet;
import com.google.android.gms.fitness.data.DataSource;
import com.google.android.gms.fitness.data.DataType;
import com.google.android.gms.fitness.data.Field;
import com.google.android.gms.fitness.request.DataReadRequest;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.chip.Chip;
import com.google.android.material.chip.ChipGroup;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
```

```
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.messaging.FirebaseMessaging;

import net.danlew.android.joda.JodaTimeAndroid;

import org.joda.time.DateTime;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Locale;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.atomic.AtomicReference;

public class MainPatientActivity extends AppCompatActivity implements
TimePickerDialog.OnTimeSetListener {

    private static final int REQUEST_OAUTH_REQUEST_CODE = 0x1001;
    private GoogleSignInClient googleSignInClient;
    private static final String TAG = "MainActivity";
    private TextView name;
    private TextView counter;
    private TextView calorie;
    private TextView Hrate;
    private TextView sleep;
    private TextView weekCounter;
    private HealthDataStorage healthDataStorage;
    private DatabaseReference heartRateReference;
    private FirebaseAuth firebaseAuth;
    ImageView setting;
    ImageView notification;
    ImageView home;
    //    medicine list for user with id = user_id;
    DatabaseReference databaseRef;

    public RecyclerView medList;
    //public TextView medUserName;
    public MedicineListAdapter medListAdapter;
    ArrayList<Medicine> list = new ArrayList<>();
    public FloatingActionButton addMedbtn;
    //public int user_id = 1;
    Button medTime;
    EditText medName, medQty;
    Switch isRepeat;
    ChipGroup chipGroup;
    Chip sun, mon, tue, wed, thu, fri, sat;

    private TextView pairingRequestText;
    private String currentPairingCode;
    private String currentCaregiverId;
    private String currentStatus;

    DatabaseReference pairingRequestRef;
    private final String postUrl =
"https://fcm.googleapis.com/v1/projects/myproject-b5ae1/messages:send";
```

```
    // Create DataSource based on Google Fit estimation of steps
    // See: https://developers.google.com/fit/scenarios/read-daily-step-
total
    static DataSource ESTIMATED_STEP_DELTAS = new DataSource.Builder()
            .setDataType(DataType.TYPE_STEP_COUNT_DELTA)
            .setType(DataSource.TYPE_DERIVED)
            .setStreamName("estimated_steps")
            .setAppPackageName("com.google.android.gms")
            .build();

    // Create a data source for calorie
    DataSource dataSourcee = new DataSource.Builder()
            //.setAppPackageName(context)
            .setDataType(DataType.TYPE_CALORIES_EXPENDED)
            .setStreamName(TAG + " - Calorie")
            .setType(DataSource.TYPE_RAW)
            .build();

    //Create datasource for heart rate
    DataSource dataSource = new DataSource.Builder()
            .setType(DataSource.TYPE_DERIVED)
            .setDataType(DataType.TYPE_HEART_RATE_BPM)
            .setAppPackageName("com.google.android.gms")
            .setStreamName("resting_heart_rate<-merge_heart_rate_bpm")
            .build();

    //Create datasource for sleep

    @SuppressLint("SetTextI18n")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_patient_main);

        // Initialise JodaTime
        JodaTimeAndroid.init(this);

        name = findViewById(R.id.textView);
        pairingRequestText = findViewById(R.id.pairingRequestText);
        counter = findViewById(R.id.counter);
        calorie = findViewById(R.id.calorie);
        Hrate = findViewById(R.id.heartBPM);
        sleep = findViewById(R.id.sleep);
        setting = findViewById(R.id.setting);
        notification = findViewById(R.id.notification);
        home = findViewById(R.id.home);
        //weekCounter = findViewById(R.id.week_counter);

        //Notification option icon
        notification.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainPatientActivity.this,
NotificationActivity.class);
                startActivity(intent);
                Log.d(TAG, "Intent: " + intent);
```

```
            }
        });

        //Setting option icon
        setting.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainPatientActivity.this,
SettingActivity.class);
                startActivity(intent);
                Log.d(TAG, "Intent: " + intent);
            }
        });

        // Initialize Google Sign-In options
        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.google_web_client_id))
                .requestEmail()
                .requestScopes(new
Scope("https://www.googleapis.com/auth/fitness.activity.read"),
                               new
Scope("https://www.googleapis.com/auth/fitness.activity.write"))
                .build();

        // Build GoogleSignInClient with the options specified by gso
        googleSignInClient = GoogleSignIn.getClient(this, gso);

        // Check for Fit permissions and retrieve health data if
available
        if (hasFitPermission()) {
            /// readStepCountDelta();
            /// readCalorieExpended(); //CALORIE
            //readHistoricStepCount();
            /// readHeartRateBPM();
            //readSleepSegment();
            readAllHealthData();
        } else {
            requestFitnessPermission();
        }

        // Retrieve the username from the intent extras
        String username = getIntent().getStringExtra("USERNAME");
        if (username == null) {
            Log.e(TAG, "USERNAME is null");
            // Handle the case where username is null
            return;
        }
        Log.d(TAG, "Retrieved username: " + username);
        name.setText("Hello, " + username);
        healthDataStorage = new HealthDataStorage(username);

        // Store the username in SharedPreferences
        SharedPreferences sharedPreferences =
getSharedPreferences("MyAppPrefs", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("username", username);
        editor.apply();
```

```
        //FOR RECYCLERVIEW
        medList = findViewById(R.id.med_list); //RecyclerView
        medList.setLayoutManager(new LinearLayoutManager(this));

        FirebaseRecyclerOptions<Medicine> options =
                new FirebaseRecyclerOptions.Builder<Medicine>()

.setQuery(FirebaseDatabase.getInstance().getReference().child("users").ch
ild(username).child("medicine"), Medicine.class)
                        .build();

        medListAdapter = new MedicineListAdapter(options, username);
        medList.setAdapter((medListAdapter));

        //Add new medicine
        databaseRef =
FirebaseDatabase.getInstance().getReference("users").child(username);
        addMedbtn = findViewById(R.id.addMed_btn); //FloatingActionButton
        addMedbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //startActivity(new Intent());
                medicineAdder(username).show();
            }
        });

        //For Pairing Between Caregiver & Patient
        pairingRequestRef =
FirebaseDatabase.getInstance().getReference("pairingRequests").child(user
name);
        addPairingListener(pairingRequestRef);

        Log.d(TAG, "DatabaseReference: " + pairingRequestRef);

        //FCMtoken
        getFCMToken(username);
        //Notification permission
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            if (ContextCompat.checkSelfPermission(this,
android.Manifest.permission.POST_NOTIFICATIONS) !=
PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.POST_NOTIFICATIONS}, 101);
            }
        }
        //Trigger notification for testing
        /*float testHeartRateValue = 110.0f; // Example heart rate value
        //notifyCaregiverAbnormalHeartRate(testHeartRateValue);
        notifyUserAbnormalHeartRate(testHeartRateValue);
        fetchPairedCaregiversAndNotify(username, "Abnormal Heart Rate
Detected", "Patient's heart rate is  "+ testHeartRateValue+" BPM. Please
check on them.");
        */
    }

    void getFCMToken(String username) {
```

```
FirebaseMessaging.getInstance().getToken().addOnCompleteListener(task ->
{
            if (task.isSuccessful()) {
                String token = task.getResult();
                Log.i("My token", token);

                // Store token in Firebase Realtime Database
                storeTokenInDatabase(username, token);
            } else {
                Log.e("Firebase", "Failed to get FCM token",
task.getException());
            }
        });
    }

    private void storeTokenInDatabase(String username, String token) {
        DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference();
        DatabaseReference tokenRef =
databaseReference.child("users").child(username).child("fcmToken");

        tokenRef.setValue(token).addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                Log.i("Firebase", "FCM token stored successfully");
            } else {
                Log.e("Firebase", "Failed to store FCM token",
task.getException());
            }
        });
    }

    private void addPairingListener(DatabaseReference pairingRequestRef)
{
        // [START post_value_event_listener]
        pairingRequestRef.addValueEventListener(new ValueEventListener()
{
            //ValueEventListener postListener = new ValueEventListener()
{
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                Log.d("DataSnapshot", "Value: " +
dataSnapshot.getValue());
                Log.d("Patient", "onDataChange called");

                for (DataSnapshot snapshot : dataSnapshot.getChildren())
{
                    PairingRequest request =
snapshot.getValue(PairingRequest.class);
                    Log.d("Patient", "Received request: " + request);
                    if (request != null &&
"pending".equals(request.getStatus())) {
                        currentPairingCode = request.getCode();
                        currentCaregiverId = request.getCaregiverId();
                        currentStatus = request.getStatus();

                        // Get the key and ensure it's not null
```

```
                        String key = snapshot.getKey();
                        if (key != null) {
                            Log.d("Key", "Key: " + key);
                            DatabaseReference pairKey =
pairingRequestRef.child(key);
                            showPairingRequestPopup(request, pairKey);
                        } else {
                            Log.w("PairingRequest", "Received null key");
                        }
                    } else {
                        Log.w("PairingRequest", "Received null request");
                    }
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError)
{
                Log.w(TAG, "loadPost:onCancelled",
databaseError.toException());
            }
        });
    }

    private void showPairingRequestPopup(PairingRequest request,
DatabaseReference pairKey) {
        pairingRequestText.setVisibility(View.VISIBLE);
        pairingRequestText.setText("Caregiver " +
request.getCaregiverId() + " wants to pair with you.");

        pairingRequestText.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(MainPatientActivity.this);
                builder.setTitle("Pairing Request")
                        .setMessage("You have received a pairing request
from " + request.getCaregiverId() + ". Do you accept?")
                        .setPositiveButton("Accept", (dialog, which) ->
showCodeInputDialog(pairKey))
                        .setNegativeButton("Reject", (dialog, which) ->
dialog.dismiss())
                        .show();
            }
        });
    }

    @SuppressLint("SetTextI18n")
    private void showCodeInputDialog(DatabaseReference pairKey) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Enter Pairing Code");

        final EditText input = new EditText(this);
        builder.setView(input);

        builder.setPositiveButton("Submit", (dialog, which) -> {
            String enteredCode = input.getText().toString().trim();
```

```
            if (enteredCode.equals(currentPairingCode)) {
                // Code matches, update status to "succeed"
                currentStatus = "succeed";
                Log.d(TAG, "Pairing status changed to: " +
currentStatus); // Log status change

                // Retrieve the username from intent extras
                String username = getIntent().getStringExtra("USERNAME");
                assert username != null;
                DatabaseReference requestRef =
FirebaseDatabase.getInstance().getReference("pairingRequests").child(user
name);

                requestRef.addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                        if (dataSnapshot.exists()) {
                            for (DataSnapshot snapshot :
dataSnapshot.getChildren()) {
                                PairingRequest request =
snapshot.getValue(PairingRequest.class);
                                if (request != null &&
"pending".equals(request.getStatus())) {
                                    request.setStatus("succeed"); //
Update status using setStatus

pairKey.setValue(request).addOnCompleteListener(task -> {
                                        if (task.isSuccessful()) {
                                            Log.d(TAG, "Pairing request
status updated to 'succeed'");
                                        } else {
                                            Log.e(TAG, "Failed to update
pairing request status", task.getException());
                                        }
                                    });
                                }
                            }
                        } else {
                            Log.e(TAG, "Pairing request not found");
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {
                        Log.e(TAG, "Failed to read pairing request",
databaseError.toException());
                    }
                });
                requestRef =
FirebaseDatabase.getInstance().getReference("pairingRequests").child(user
name).child(currentCaregiverId);
                requestRef.child("status").setValue("approved");
```

```
                DatabaseReference caregiverRef =
FirebaseDatabase.getInstance().getReference("users").child(currentCaregiv
erId).child("pairedPatients").child(username);
                caregiverRef.setValue(true);

                DatabaseReference patientRef =
FirebaseDatabase.getInstance().getReference("users").child(username).chil
d("pairedCaregivers").child(currentCaregiverId);
                patientRef.setValue(true);

                Toast.makeText(MainPatientActivity.this, "Pairing with "
+ currentCaregiverId + " successful!", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(MainPatientActivity.this, "Invalid code.
Please try again.", Toast.LENGTH_SHORT).show();
                showCodeInputDialog(pairKey);
            }
        });

        builder.setNegativeButton("Cancel", (dialog, which) ->
dialog.cancel());

        builder.show();
    }

    @Override
    protected void onStart() {
        super.onStart();
        medListAdapter.startListening();
    }

    @Override
    protected void onStop() {
        super.onStop();
        medListAdapter.stopListening();
    }

    /**
     * Request Fitness permission of the user. This process will present
an account dialog for the
     * user to select their Google account, and then the Fitness
permissions dialog.
     */
    private void requestFitnessPermission() {
        GoogleSignIn.requestPermissions(
                this,
                REQUEST_OAUTH_REQUEST_CODE,
                GoogleSignIn.getLastSignedInAccount(this),
                getFitnessSignInOptions());
    }

    /**
     * Verify if the app has permissions to fetch Fitness data.
     *
     * @return true if user has permitted permissions.
     */
    private boolean hasFitPermission() {
        // Request permission to collect Google Fit data
```

```
        FitnessOptions fitnessOptions = getFitnessSignInOptions();
        return
GoogleSignIn.hasPermissions(GoogleSignIn.getLastSignedInAccount(this),
fitnessOptions);
    }

    /**
     * Specify which data types we would like access to. This is
presented to the user as a list
     * of permissions we are seeking to get approved.
     *
     * @return the FitnessOptions containing the data types.
     */
    private FitnessOptions getFitnessSignInOptions() {
        // Request access to step count data from Fit history
        return FitnessOptions.builder()
                .addDataType(DataType.TYPE_STEP_COUNT_CUMULATIVE)
                .addDataType(DataType.TYPE_STEP_COUNT_DELTA)
                .addDataType(DataType.TYPE_CALORIES_EXPENDED)
                .addDataType(DataType.TYPE_HEART_RATE_BPM)
                .addDataType(DataType.TYPE_SLEEP_SEGMENT)
                .build();
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        // When the user has accepted the use of Fit data,
subscribeStepCount to record data
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {
            if (requestCode == REQUEST_OAUTH_REQUEST_CODE) {
                Log.i(TAG, "Fitness permission granted");
                subscribeStepCount();
                //readStepCountDelta(); // Read today's data
                //readHistoricStepCount(); // Read last weeks data//mmg
comment dhlama
                //readCalorieExpended(); //Read calorie
                //readHeartRateBPM(); //Read heart rate
                readSleepSegment(); //Read sleep
                readAllHealthData();
            }
        } else {
            Log.i(TAG, "Fitness permission denied");
        }
    }

    /**
     * Request a subscription to record step data on the background. This
means that the app will
     * record the step count and push it to the Fitness history. Without
this, the Google Fit app
     * must be installed to do the recording for us!
     */
    private void subscribeStepCount() {
        // To create a subscription, invoke the Recording API. As soon as
the subscription is
```

```
        // active, fitness data will start recording.
        Fitness.getRecordingClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .subscribe(DataType.TYPE_STEP_COUNT_CUMULATIVE);
    }

    private void readAllHealthData() {
        String username = getIntent().getStringExtra("USERNAME");
        if (!hasFitPermission()) {
            requestFitnessPermission();
            return;
        }

        // Initialize data variables
        final AtomicInteger stepCount = new AtomicInteger(0);
        final AtomicReference<Float> calorieExpenditure = new
AtomicReference<>(0f);
        final AtomicReference<Float> heartRate = new
AtomicReference<>(0f);
        final AtomicInteger remainingTasks = new AtomicInteger(3);

        // Success callback to save data when all data is fetched
        Runnable saveDataIfComplete = () -> {
            if (remainingTasks.decrementAndGet() == 0) {
                healthDataStorage.saveHealthData(stepCount.get(),
calorieExpenditure.get(), heartRate.get());

                // Check if heart rate or step count is empty and notify
if necessary
                if (heartRate.get() == 0 || stepCount.get() == 0) {
                    notifyUserToWearSmartwatch();
                    fetchPairedCaregiversAndNotify(username, "Patient
Missing Wearable", "Patient "+ username +" is not wearing smartwatch.
Please remind them to wear.");
                }
            }
        };

        // Read step count
        Fitness.getHistoryClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .readDailyTotal(DataType.AGGREGATE_STEP_COUNT_DELTA)
                .addOnSuccessListener(dataSet -> {
                    stepCount.set(dataSet.isEmpty() ? 0 :
dataSet.getDataPoints().get(0).getValue(Field.FIELD_STEPS).asInt());
                    saveDataIfComplete.run();
                    counter.setText(String.format(Locale.ENGLISH, "%d",
stepCount.get()));
                })
                .addOnFailureListener(e -> Log.w(TAG, "There was a
problem getting the step count.", e));

        // Read calorie expended
        Fitness.getHistoryClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .readDailyTotal(DataType.AGGREGATE_CALORIES_EXPENDED)
                .addOnSuccessListener(dataSet -> {
                    if (!dataSet.isEmpty()) {
```

```
                        calorieExpenditure.set((float)
Math.round(dataSet.getDataPoints().get(0).getValue(Field.FIELD_CALORIES).
asFloat()));
                    }
                    calorie.setText(String.format(Locale.ENGLISH, "%d",
calorieExpenditure.get().intValue()));
                    saveDataIfComplete.run();
                })
                .addOnFailureListener(e -> Log.w(TAG, "There was a
problem getting the calorie.", e));

        // Read heart rate
        long endTime = System.currentTimeMillis();
        long startTime = endTime - TimeUnit.DAYS.toMillis(7);
        DataReadRequest readRequest = new DataReadRequest.Builder()
                .read(DataType.TYPE_HEART_RATE_BPM)
                .setTimeRange(startTime, endTime, TimeUnit.MILLISECONDS)
                .build();
        Fitness.getHistoryClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .readData(readRequest)
                .addOnSuccessListener(dataReadResponse -> {
                    float maxNormalHeartRate = 100; // Example: Define
your max normal heart rate
                    float minNormalHeartRate = 60;  // Example: Define
your min normal heart rate

                    Log.d(TAG, "DataReadResponse received");

                    for (DataSet dataSet :
dataReadResponse.getDataSets()) {
                        Log.d(TAG, "DataSet: " + dataSet);

                        for (DataPoint dataPoint :
dataSet.getDataPoints()) {
                            float heartRateValue =
dataPoint.getValue(Field.FIELD_BPM).asFloat();
                            Log.d(TAG, "Heart rate value: " +
heartRateValue);

                            // Check for abnormal heart rate
                            if (heartRateValue < minNormalHeartRate ||
heartRateValue > maxNormalHeartRate) {
                                // Notify user about abnormal heart rate
                                fetchPairedCaregiversAndNotify(username,
"Abnormal Heart Rate Detected", "Patient's heart rate is " +
heartRateValue + " BPM. Please check on them.");

notifyUserAbnormalHeartRate(heartRateValue);
                            }
                            // Update the UI with the latest heart rate
value
                            heartRate.set(heartRateValue);
                        }
                    }

                    // Save or display heart rate data as required
                    String heartRateText = heartRate.get() + " BPM";
```

```
                    Hrate.setText(heartRateText);
                    saveDataIfComplete.run();
                })
                .addOnFailureListener(e -> Log.w(TAG, "There was a
problem getting the heart rate.", e));
    }

    private void readSleepSegment() {
        if (!hasFitPermission()) {
            requestFitnessPermission();
            return;
        }

        Fitness.getHistoryClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .readDailyTotal(DataType.TYPE_SLEEP_SEGMENT)
                .addOnSuccessListener(
                        new OnSuccessListener<DataSet>() {
                            @Override
                            public void onSuccess(DataSet dataSet) {
                                long total =
                                        dataSet.isEmpty()
                                                ? 0
                                                :
dataSet.getDataPoints().get(0).getValue(Field.FIELD_SLEEP_SEGMENT_TYPE).a
sInt();
                                /*if (!dataSet.isEmpty()) {
                                    total =
Math.round(dataSet.getDataPoints().get(0).getValue(Field.FIELD_SLEEP_SEGM
ENT_TYPE).asInt());
                                }*/

sleep.setText(String.format(Locale.ENGLISH, "%d", total));
                                healthDataStorage.saveSleepData((int)
total); //To save data in database
                            }
                        })
                .addOnFailureListener(
                        new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Log.w(TAG, "There was a problem getting
the sleep data.", e);
                            }
                        });
    }

    private AlertDialog medicineAdder(String username) {


        View layout = View.inflate(this, R.layout.add_med_dialog, null);
//        medicine details:
        medName = layout.findViewById(R.id.add_med_name);
        medQty = layout.findViewById(R.id.add_med_qty);
        medTime = layout.findViewById(R.id.add_med_time);
//        UI components:
        isRepeat = layout.findViewById(R.id.repeat_switch);
        chipGroup = layout.findViewById(R.id.chip_group);
```

```
        setChildrenEnabled(chipGroup, false);
        sun = layout.findViewById(R.id.sunday);
        mon = layout.findViewById(R.id.monday);
        tue = layout.findViewById(R.id.tuesday);
        wed = layout.findViewById(R.id.wednesday);
        thu = layout.findViewById(R.id.thursday);
        fri = layout.findViewById(R.id.friday);
        sat = layout.findViewById(R.id.saturday);

        medTime.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DialogFragment timePicker = new TimePickerFragment();
                timePicker.show(getSupportFragmentManager(), "time
picker");
            }
        });

        isRepeat.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!isRepeat.isChecked()) {
                    setChildrenEnabled(chipGroup, false);
                } else {
                    setChildrenEnabled(chipGroup, true);
                }
            }
        });


        AlertDialog.Builder builder = new AlertDialog.Builder(this);

        builder.setPositiveButton("ADD", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String temp = medQty.getText().toString();
                int qty = 0;
                if (!"".equals(temp))
                    qty = Integer.parseInt(temp);
                String days = "0000000";
                if (isRepeat.isChecked()) {
                    days = setDaysFormat(sun, mon, tue, wed, thu, fri,
sat);
                }

                // Add medicine data to Firebase Realtime Database
                String medicineKey =
databaseRef.child("medicine").push().getKey(); // Generate a unique key
for the medicine entry
                HashMap<String, Object> medicineData = new HashMap<>();
                medicineData.put("medName",
medName.getText().toString());
                medicineData.put("qty", qty);
                medicineData.put("medTime",
medTime.getText().toString());
                medicineData.put("days", days);
```

```
databaseRef.child("medicine").child(medicineKey).setValue(medicineData);

                // Add the new medicine to the local list
                Medicine newMedicine = new
Medicine(medName.getText().toString(), qty, medTime.getText().toString(),
days);
                list.add(newMedicine);

                // Notify the adapter of the change
                medListAdapter.notifyDataSetChanged();

            }
        });

        builder.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        builder.setView(layout);
        return builder.create();
    }

    public String setDaysFormat(Chip sun, Chip mon, Chip tue, Chip wed,
Chip thu, Chip fri, Chip sat) {
        String dayString = "" + (sun.isChecked() ? "1" : "0") +
(mon.isChecked() ? "1" : "0") + (tue.isChecked() ? "1" : "0") +
(wed.isChecked() ? "1" : "0") + (thu.isChecked() ? "1" : "0") +
(fri.isChecked() ? "1" : "0") + (sat.isChecked() ? "1" : "0");
        return dayString;
    }

    public void setChildrenEnabled(ChipGroup chipGroup, Boolean enable) {
        for (int i = 0; i < chipGroup.getChildCount(); i++) {
            chipGroup.getChildAt(i).setEnabled(enable);
        }

    }

    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        medTime.setText(hourOfDay + ":" + minute);
    }

    //NOTIFICATION SECTION
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == 101) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                // Permission granted
            } else {
```

```
                // Permission denied
                Toast.makeText(this, "Notification permission is required
for this feature.", Toast.LENGTH_SHORT).show();
            }
        }
    }
    private void notifyUserAbnormalHeartRate(float heartRateValue) {
        // Define the channel details
        String channelId = "abnormal_heart_rate_channel";
        String channelName = "Abnormal Heart Rate Alerts";
        String channelDescription = "Notifications for abnormal heart
rate";
        int importance = NotificationManager.IMPORTANCE_HIGH;

        // Create the notification channel
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new
NotificationChannel(channelId, channelName, importance);
            channel.setDescription(channelDescription);
            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }

        // Create the notification content
        String message = String.format(Locale.ENGLISH, "Your current
heart rate is %.1f BPM, which is abnormal.", heartRateValue);

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, channelId)
                .setSmallIcon(R.drawable.ic_1) // Set your app's icon
here
                .setContentTitle("Abnormal Heart Rate Detected")
                .setContentText(message)
                .setAutoCancel(true)
                .setPriority(NotificationCompat.PRIORITY_HIGH);

        Intent intent = new Intent(this, MainPatientActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
intent, PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE);
        builder.setContentIntent(pendingIntent);

        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(0, builder.build());
    }

    private void notifyUserToWearSmartwatch() {
        String channelId = "wear_smartwatch_channel";
        String channelName = "Smartwatch Reminder";
        String message = "Please wear your smartwatch to track your
health data.";
        String channelDescription = "Notifications for missing
wearables";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
```

```
        // Create the notification channel
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new
NotificationChannel(channelId, channelName, importance);
            channel.setDescription(channelDescription);
            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel);
        }

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, channelId)
                .setSmallIcon(R.drawable.notification) // Set your app's
icon here
                .setContentTitle("Reminder")
                .setContentText(message)
                .setAutoCancel(true)
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);

        Intent intent = new Intent(this, NotificationActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
intent, PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE);
        builder.setContentIntent(pendingIntent);

        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(0, builder.build());
    }

    private void notifyCaregiverAbnormalHeartRate(float heartRateValue) {
        //for testing notification
        FcmNotificationsSender fcmNotificationsSender = new
FcmNotificationsSender(

"fOGNsSxUTWuCGkYOd_okot:APA91bEFDpEnHpzmpU_iXLjorrK9Xi3n_IHICYM_vzki_G-
Fv0zBv8oj0ad3dbcPnL04lUxjkGxwlq1WCXU95Q7tkxAz9yroG1PYhZHzxdyu81VotC1Z4yDr
DVNxY_B5QGwMxgoADhlO",
                "Abnormal Heart Rate Detected", "Patient's heart rate is
" + heartRateValue + " BPM. Please check on them.",
                 getApplicationContext()
        );

        fcmNotificationsSender.SendNotifications();

    }

    private void notifyCaregiverAboutUser(String username) {
        //for testing notification
        FcmNotificationsSender fcmNotificationsSender = new
FcmNotificationsSender(

"fOGNsSxUTWuCGkYOd_okot:APA91bEFDpEnHpzmpU_iXLjorrK9Xi3n_IHICYM_vzki_G-
Fv0zBv8oj0ad3dbcPnL04lUxjkGxwlq1WCXU95Q7tkxAz9yroG1PYhZHzxdyu81VotC1Z4yDr
DVNxY_B5QGwMxgoADhlO",
```

```
                "Patient Missing Wearable", "Patient "+ username +"is not
wearing smartwatch. Please remind them to wear.",
                getApplicationContext()
        );

        fcmNotificationsSender.SendNotifications();

    }

//   FETCHING FCM TOKENN ////////////////////
private void fetchPairedCaregiversAndNotify(String patientUsername,
String notificationTitle, String notificationMessage) {
    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();
    DatabaseReference pairedCaregiversRef =
databaseRef.child("users").child(patientUsername).child("pairedCaregivers
");

    pairedCaregiversRef.addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot caregiverSnapshot :
dataSnapshot.getChildren()) {
                String caregiverUsername = caregiverSnapshot.getKey();
                Log.d(TAG, "Processing caregiver username: " +
caregiverUsername);
                if (caregiverUsername != null) {
                    fetchCaregiverFcmToken(caregiverUsername,
notificationTitle, notificationMessage);
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Log.w(TAG, "Failed to retrieve paired caregivers.",
databaseError.toException());
        }
    });
}


    private void fetchCaregiverFcmToken(String caregiverUsername, String
title, String message) {
        DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();
        DatabaseReference fcmTokenRef =
databaseRef.child("users").child(caregiverUsername).child("fcmToken");

        fcmTokenRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
{
                String fcmToken = dataSnapshot.getValue(String.class);
                Log.d(TAG, "Retrieved FCM token for caregiver " +
caregiverUsername + ": " + fcmToken);
```

```
                if (fcmToken != null) {
                    sendNotification(fcmToken, title, message);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError)
{
                Log.w(TAG, "Failed to retrieve FCM token for caregiver.",
databaseError.toException());
            }
        });
    }

    //Sending Notification
    private void sendNotification(String fcmToken, String title, String
message) {
        FcmNotificationsSender fcmNotificationsSender = new
FcmNotificationsSender(
                fcmToken,
                title,
                message,
                getApplicationContext()
        );
        fcmNotificationsSender.SendNotifications();
    }

//////////////////////END OF FETCHING
////////////////////////////////////
    /**
     * Asynchronous task to read the history data. When the task
succeeds, it will print out the data.
     */
    /*private void readHistoricStepCount() {
        if (!hasFitPermission()) {
            requestFitnessPermission();
            return;
        }

        // Invoke the History API to fetch the data with the query
        Fitness.getHistoryClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .readData(queryFitnessData())
                .addOnSuccessListener(
                        new OnSuccessListener<DataReadResponse>() {
                            @Override
                            public void onSuccess(DataReadResponse
dataReadResponse) {
                                // For the sake of the sample, we'll
print the data so we can see what we just
                                // added. In general, logging fitness
information should be avoided for privacy
                                // reasons.
                                printData(dataReadResponse);
                            }
                        })
                .addOnFailureListener(
                        new OnFailureListener() {
```

```
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Log.e(TAG, "There was a problem reading
the historic data.", e);
                            }
                        });
    }*/

    /**
     * Returns a {@link DataReadRequest} for all step count changes in
the past week.
     */
    public static DataReadRequest queryFitnessData() {
        // [START build_read_data_request]
        // Setting a start and end date using a range of 1 week working
backwards using today's
        // start of the day (midnight). This ensures that the buckets are
in line with the days.
        DateTime dt = new DateTime().withTimeAtStartOfDay();
        long endTime = dt.getMillis();
        long startTime = dt.minusWeeks(1).getMillis();

        return new DataReadRequest.Builder()
                // The data request can specify multiple data types to
return, effectively
                // combining multiple data queries into one call.
                // In this example, it's very unlikely that the request
is for several hundred
                // datapoints each consisting of a few steps and a
timestamp.  The more likely
                // scenario is wanting to see how many steps were walked
per day, for 7 days.
                .aggregate(ESTIMATED_STEP_DELTAS,
DataType.AGGREGATE_STEP_COUNT_DELTA)
                // Analogous to a "Group By" in SQL, defines how data
should be aggregated.
                // bucketByTime allows for a time span, whereas
bucketBySession would allow
                // bucketing by "sessions", which would need to be
defined in code.
                .bucketByTime(1, TimeUnit.DAYS)
                .setTimeRange(startTime, endTime, TimeUnit.MILLISECONDS)
                .build();
    }

    /*public void printData(DataReadResponse dataReadResult) {
        StringBuilder result = new StringBuilder();
        // [START parse_read_data_result]
        // If the DataReadRequest object specified aggregated data,
dataReadResult will be returned
        // as buckets containing DataSets, instead of just DataSets.
        if (dataReadResult.getBuckets().size() > 0) {
            Log.i(TAG, "Number of returned buckets of DataSets is: " +
dataReadResult.getBuckets().size());
            for (Bucket bucket : dataReadResult.getBuckets()) {
                List<DataSet> dataSets = bucket.getDataSets();
                for (DataSet dataSet : dataSets) {
                    result.append(formatDataSet(dataSet));
```

```
                }
            }
        } else if (dataReadResult.getDataSets().size() > 0) {
            Log.i(TAG, "Number of returned DataSets is: " +
dataReadResult.getDataSets().size());
            for (DataSet dataSet : dataReadResult.getDataSets()) {
                result.append(formatDataSet(dataSet));
            }
        }
        // [END parse_read_data_result]
        weekCounter.setText(result);
    }*/

    /**
     * Format the dataset value to a string; Mon: 1000
     *
     * @param dataSet
     * @return formatted string
     */
    private static String formatDataSet(DataSet dataSet) {
        StringBuilder result = new StringBuilder();

        for (DataPoint dp : dataSet.getDataPoints()) {
            // Get the day of the week JodaTime property
            DateTime sDT = new
DateTime(dp.getStartTime(TimeUnit.MILLISECONDS));
            DateTime eDT = new
DateTime(dp.getEndTime(TimeUnit.MILLISECONDS));

            result.append(
                    String.format(
                            Locale.ENGLISH,
                            "%s %s to %s %s\n",
                            sDT.dayOfWeek().getAsShortText(),
                            sDT.toLocalTime().toString("HH:mm"),
                            eDT.dayOfWeek().getAsShortText(),
                            eDT.toLocalTime().toString("HH:mm")
                    )
            );

            result.append(
                    String.format(
                            Locale.ENGLISH,
                            "%s: %s %s\n",
                            sDT.dayOfWeek().getAsShortText(),

dp.getValue(dp.getDataType().getFields().get(0)).toString(),

dp.getDataType().getFields().get(0).getName()));
        }

        return String.valueOf(result);
    }


    /*@Override
    public boolean onOptionsItemSelected(final MenuItem item) {
        int id = item.getItemId();
```

```
        if (id == R.id.action_revoke) {
            revokeFitnessPermissions();
        }
        if (id == R.id.action_read_data) {
            readStepCountDelta();
            return true;
        } else if (id == R.id.action_read_historic_data) {
            readHistoricStepCount();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }*/

    private void revokeFitnessPermissions() {
        if (!hasFitPermission()) {
            // No need to revoke if we don't already have permissions
            return;
        }

        // Stop recording the step count
        Fitness.getRecordingClient(this,
GoogleSignIn.getLastSignedInAccount(this))
                .unsubscribe(DataType.TYPE_STEP_COUNT_CUMULATIVE);

        // Revoke Fitness permissions
        GoogleSignInOptions signInOptions = new
GoogleSignInOptions.Builder().addExtension(getFitnessSignInOptions()).bui
ld();
        GoogleSignIn.getClient(this, signInOptions).revokeAccess();

        Toast.makeText(this, "Fitness permissions revoked",
Toast.LENGTH_SHORT).show();
    }

}
```