```java
package com.example.covid_19alertapp.activities;


import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.lifecycle.MutableLiveData;

import androidx.lifecycle.Observer;


import android.content.DialogInterface;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;


import com.example.covid_19alertapp.R;

import com.example.covid_19alertapp.extras.Constants;

import com.example.covid_19alertapp.extras.LogTags;

import com.example.covid_19alertapp.models.InfectedLocations;

import com.example.covid_19alertapp.roomdatabase.LocalDBContainer;

import com.example.covid_19alertapp.roomdatabase.VisitedLocations;

import com.example.covid_19alertapp.roomdatabase.VisitedLocationsDao;

import com.example.covid_19alertapp.roomdatabase.VisitedLocationsDatabase;
import com.example.covid_19alertapp.sharedPreferences.MiscSharedPreferences;

import com.example.covid_19alertapp.sharedPreferences.UserInfoSharedPreferences;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseException;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;


import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;


public class UploadLocationsActivity extends AppCompatActivity {

/*

upload locations from local db to firebase

implement verification by medical report photo here

 */


    // firebase

    //private FirebaseDatabase firebaseDatabase;

    private DatabaseReference firbaseReference;


    // local db

    private VisitedLocationsDatabase roomDatabase;
    private VisitedLocationsDao visitedLocationsDao;


    // retrieved data from local db
    private List<VisitedLocations> retrievedDatas = new ArrayList<>();


    // retrieve and upload progress level
    private int dataSize, dataCount = 0;
    private double currProgress = 0;


    // models to store in firebase
    private MutableLiveData<InfectedLocations> currentInfectedLocation = new MutableLiveData<>();
    final Observer<InfectedLocations> newEntryObserver = new Observer<InfectedLocations>() {
        @Override
        public void onChanged(final InfectedLocations infectedLocations) {


            if(!infectedLocations.allFieldsSet()) {
                // exit if all values not set
                Log.d(LogTags.Upload_TAG, "onChanged: all fields not set");
                return;
            }


            // upload to firebase
            insertToFirebase("infectedLocations", infectedLocations.getKey(),
  infectedLocations.getDateTime(), infectedLocations.getCount());


            // blacklist user
            // get user uid
            String uid = UserInfoSharedPreferences.getUid(UploadLocationsActivity.this);


            insertToFirebase("blackList/"+uid+"/visitedLocations",
                    infectedLocations.getKey(), infectedLocations.getDateTime(), infectedLocations.getCount());


        }
    };


    // UI stuff
    ProgressBar uploadProgressBar;
    TextView uploadProgressText;
    Button uploadButton, home_btn;


    // back press during uploading
    boolean uploading = false;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_upload_locations);
        home_btn = findViewById(R.id.home_button_upload_locations);
        home_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

        setUpUI();

        // set firebase database offline capability, set firebase reference
        if(firbaseReference == null) {
            FirebaseDatabase database = FirebaseDatabase.getInstance();
            try {
                database.setPersistenceEnabled(true);
            }catch (DatabaseException e){
                Log.d(LogTags.Upload_TAG, "onCreate: setPersistent issue. need to fix this");
            }
            firbaseReference  = database.getReference();
        }

        // set local db configs
        roomDatabase = VisitedLocationsDatabase.getDatabase(getApplicationContext());
        visitedLocationsDao = roomDatabase.visitedLocationsDao();

        // set InfectedLocation Live Data observer
        currentInfectedLocation.observe(this, newEntryObserver);
```

```
            );


    }



    @Override
    public void onBackPressed() {


        if(uploading) {

            // show dialog
            Log.d(LogTags.Upload_TAG, "onBackPressed: back pressed during uploading");


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

            builder.setMessage(getText(R.string.backPressed_during_upload))
                    .setCancelable(false)
                    .setPositiveButton(getText(R.string.backPressed_during_upload_positive), new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                            Log.d(LogTags.Upload_TAG, "onClick: uploading resumes");
                        }
                    });
        AlertDialog alertDialog = builder.create();

        alertDialog.show();


    }

    else

        super.onBackPressed();


}


private void setUpUI() {


    uploadProgressBar = findViewById(R.id.uploadProgressBar);

    uploadProgressText = findViewById(R.id.uploadProgressText);

    uploadButton = findViewById(R.id.upload_btn);


}


private void uploadAndDeleteLocal() {

    /*

    retrive from local database,

    upload to firebase,

    delete from local databse

     */


    // save the uploading state
```

```
        uploading = true;

        uploadProgressText.setVisibility(View.VISIBLE);

        uploadProgressBar.setVisibility(View.VISIBLE);


        roomDatabase.databaseWriteExecutor.execute(new Runnable() {

            @Override

            public void run() {


                // fetch all from localDB

                retrievedDatas = visitedLocationsDao.fetchAll();

                Log.d(LogTags.Upload_TAG, "onCreate: local database retrieved");


                // retrieval from localDB done (50%)

                currProgress = 50;

                dataSize = retrievedDatas.size();




                if(dataSize==0) {

                    // notify on UI thread no data found locally

                    runOnUiThread(new Runnable() {

                        @Override

                        public void run() {

                            Toast.makeText(UploadLocationsActivity.this, "No locations recorded, only home
address uploaded", Toast.LENGTH_LONG)

                                    .show();

                            uploadProgressText.setVisibility(View.GONE);
                            uploadProgressBar.setVisibility(View.GONE);
                        }
                    });

                    uploading = false;

                    return;
                }


                for(VisitedLocations roomEntry: retrievedDatas){

                    // splitData[0] = lat,lon
                    // splitData[1] = dateTime
                    String[] splitData = roomEntry.splitPrimaryKey();

                    Log.d(LogTags.Upload_TAG, "run: current retrieved data = "
                            +splitData[0]+", "+roomEntry.getCount()+", "+splitData[1]);

                    // set the LiveData object
                    currentInfectedLocation.postValue(new InfectedLocations(splitData[0],
roomEntry.getCount(), splitData[1]));

                    // delete current entry from local database
                    visitedLocationsDao.deleteLocation(roomEntry);
                    Log.d(LogTags.Upload_TAG, "onCreate: deleting room entry = "
            +roomEntry.getConatainerDateTimeComposite());


// keep track of upload progress (50%-100%)

currProgress += (double) 50/dataSize;

uploadProgressBar.setProgress((int) currProgress);


dataCount++;

if(dataCount==dataSize){


  runOnUiThread(new Runnable() {

            @Override

            public void run() {

              // remove progressbar

              uploadProgressText.setText(getText(R.string.uploadFinished_progressbar_text));

              uploadProgressBar.setVisibility(View.GONE);

            }

  });


  // uploading done

  uploading = false;


  // set upload status shared preference true

  MiscSharedPreferences.setUploadStatus(UploadLocationsActivity.this, true);


}
```

```
                // sleep, give time to upload properly?
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    Log.d(LogTags.Upload_TAG, "run: thread just had coffee and isn't tired rn");
                    e.printStackTrace();
                }
            }

        }
    });

}

private void uploadHomeLocation(){

    List<String> entries;

    String homeLatLng = UserInfoSharedPreferences.getHomeLatLng(this);
    if(homeLatLng.equals("")){
        Log.d(LogTags.Upload_TAG, "uploadHomeLocation: why the hell is home null");
        return;
    }
```

```
        String[] latLng = homeLatLng.split(",");


        entries = LocalDBContainer.calculateContainer(Double.parseDouble(latLng[0]),
    Double.parseDouble(latLng[1]), "Bangladesh");


        // get current time

        Calendar cal = Calendar.getInstance();

        //TODO: add year

        final String dateTime = (cal.get(Calendar.MONTH)+1) +"-" // Calender.MONTH is 0 based -_- why tf?

                + cal.get(Calendar.DATE) +"-"

                + cal.get(Calendar.HOUR_OF_DAY);


        for (String entry: entries) {


            // need '@' instead of '.'

            entry = entry.replaceAll("\\.","@");


            // upload home address

            insertToFirebase("infectedHomes", entry, dateTime, 1);


            // blacklist user


            // get user uid

            String uid = UserInfoSharedPreferences.getUid(this);


            insertToFirebase("blackList/"+uid+"/home", entry, dateTime, 1);
        }

    }

    private void insertToFirebase(final String node, String latLon, String dateTime, final long count){

        final DatabaseReference currentReference =
firbaseReference.child(node).child(latLon).child(dateTime);

        currentReference.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {



                if(dataSnapshot.child("unverifiedCount").getValue()!=null){
                    // data already exists
                    Log.d(LogTags.Upload_TAG, "onDataChange: location already exists at "+node);

                    long existingCount = (long)dataSnapshot.child("unverifiedCount").getValue();

                    currentReference.child("unverifiedCount").setValue(count + existingCount);

                }

                else{
    }

            // no such data exists
            Log.d(LogTags.Upload_TAG, "onDataChange: new location at "+node);

            currentReference.child("unverifiedCount").setValue(count);
          }

          if(dataSnapshot.child("verifiedCount").getValue()==null)
            currentReference.child("verifiedCount").setValue(0);


        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

          Log.d(LogTags.Upload_TAG, "onCancelled: firebase e somossa ki korbo?
"+databaseError.getMessage() +", "+databaseError.getDetails());

          Toast.makeText(getApplicationContext(),
              getApplicationContext().getString(R.string.no_internet_toast),
              Toast.LENGTH_LONG)
              .show();

        }
      });

    }
    public void uploadClicked(View view) {
        /*
        upload button click
         */

        // show dialog before uploading

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

        builder.setTitle(getText(R.string.upload_confirmation_title))
                .setMessage(getText(R.string.upload_confirmation_message))
                .setCancelable(false)
                .setPositiveButton(getText(R.string.upload_confirmation_positive), new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();

                        Log.d(LogTags.Upload_TAG, "onClick: uploading starts");

                        // upload home location
                        uploadHomeLocation();

                        // start uploading process
                        uploadButton.setEnabled(false);

                        uploadAndDeleteLocal();




                    }

                })

                .setNegativeButton(getText(R.string.upload_confirmation_negative), new
DialogInterface.OnClickListener() {

                    @Override

                    public void onClick(DialogInterface dialog, int which) {

                        dialog.dismiss();


                        // close the activity

                        UploadLocationsActivity.this.finish();

                        Log.d(LogTags.Upload_TAG, "onClick: not gonna upload");

                    }

                });


        AlertDialog alertDialog = builder.create();

        alertDialog.show();


    }

}
```