package com.example.containmentzone\_alert.activities;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;

import android.os.Handler;

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.containmentzone\_alert.R;

import com.example.containmentzone\_alert.adapters.LocationListAdapter;

import com.example.containmentzone\_alert.extras.AddressReceiver;

import com.example.containmentzone\_alert.extras.Constants;

import com.example.containmentzone\_alert.extras.Internet;

import com.example.containmentzone\_alert.extras.LogTags;

import com.example.containmentzone\_alert.extras.LogTags;

import com.example.containmentzone\_alert.extras.Notifications;

import com.example.containmentzone\_alert.models.MatchedLocation;

import com.example.containmentzone\_alert.roomdatabase.VisitedLocations;

import com.example.containmentzone\_alert.roomdatabase.VisitedLocationsDao;

import com.example.containmentzone\_alert.roomdatabase.VisitedLocationsDatabase;

import com.example.containmentzone\_alert.sharedPreferences.UserInfoSharedPreferences;

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 java.util.ArrayList;

import java.util.List;

public class ShowMatchedLocationsActivity extends AppCompatActivity implements AddressReceiver.AddressView {

// matched locations model (for recycler-view)

ArrayList<MatchedLocation> matchedLocations = new ArrayList<>();

int matchedLocationPosition = 0, locationQueryCount = 0;

// matched home locations model (for another(?) recycler-view)

ArrayList<MatchedLocation> matchedHomeLocations = new ArrayList<>();

int homeQueryCount = 0;

// firebase

private DatabaseReference firebaseReference;

// local db

private VisitedLocationsDatabase roomDatabase;

private VisitedLocationsDao visitedLocationsDao;

// retrieved data from local db

private List<VisitedLocations> retrievedDatas = new ArrayList<>();

private int dataSize;

// Address Fetch

AddressReceiver addressReceiver = new AddressReceiver(new Handler(), this);

// UI stuff

private ProgressBar progressBar;

private TextView progressBarText;

private Button retryButton;

private RecyclerView locationRecyclerView, homeLocationRecyclerView;

private LocationListAdapter locationListAdapter, homeLocationListAdapter;

private boolean internetAvailable = true;

// flags

private boolean localDbEmptyFlag = false;

private boolean homeLocationsFetchFinishedFlag = false;

private boolean locationsFetchFinishedFlag = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_show\_matched\_locations);

setUI();

Notifications.removeNotification(Constants.DangerNotification\_ID, this);

// set local db configs

roomDatabase = VisitedLocationsDatabase.getDatabase(getApplicationContext());

visitedLocationsDao = roomDatabase.visitedLocationsDao();

// firebase

firebaseReference = FirebaseDatabase.getInstance().getReference();

findHomeMatchedLocations();

findMatchedLocations();

}

private void setUI() {

progressBar = findViewById(R.id.progressBar);

progressBarText = findViewById(R.id.progressText);

retryButton = findViewById(R.id.retry\_btn);

homeLocationRecyclerView = findViewById(R.id.homeRecyclerView);

homeLocationRecyclerView.setLayoutManager(new LinearLayoutManager(this));

locationRecyclerView = findViewById(R.id.locationRecyclerView);

locationRecyclerView.setLayoutManager(new LinearLayoutManager(this));

}

private void findHomeMatchedLocations() {

homeLocationsFetchFinishedFlag = false;

matchedHomeLocations.clear();

homeQueryCount = 0;

homeLocationListAdapter = new LocationListAdapter(this, matchedHomeLocations);

homeLocationRecyclerView.setAdapter(homeLocationListAdapter);

List<String> queryKeys;

final String homeLatLng = UserInfoSharedPreferences.getHomeLatLng(this);

if(homeLatLng.equals("")){

Log.d(LogTags.Worker\_TAG, "queryHomeAddress: why the hell is home null");

return;

}

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

queryKeys =

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

final int querySize = queryKeys.size();

for (String query: queryKeys) {

if(!Internet.isInternetAvailable(getApplicationContext())){

runOnUiThread(new Runnable() {

@Override

public void run() {

internetDisconncetedUI();

}

});

return;

}

// need '@' instead of '.'

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

firebaseReference.child("infectedHomes").child(query)

.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.getValue()!=null){

long verifiedCount = 0, unverifiedCount = 0;

for (DataSnapshot snapshot: dataSnapshot.getChildren()) {

verifiedCount+=(long)snapshot.child("verifiedCount").getValue();

unverifiedCount+=(long) snapshot.child("unverifiedCount").getValue();

}

MatchedLocation homeLocation = new MatchedLocation(

Double.parseDouble(latLng[0]),

Double.parseDouble(latLng[1]),

"NEAR YOUR HOME!",

verifiedCount,

unverifiedCount

);

if(matchedHomeLocations.isEmpty()) {

// only find one match for home

matchedHomeLocations.add(homeLocation);

homeLocationListAdapter.notifyItemInserted(matchedHomeLocations.size() - 1);

homeLocationsFetchFinishedFlag = true;

if(locationsFetchFinishedFlag)

dataFetchFinishedUI();

else if(localDbEmptyFlag)

localDbEmptyUI();

}

Log.d(LogTags.MatchFound\_TAG, "onDataChange: home location matched: "+homeLocation.toString());

}

homeQueryCount++;

if(homeQueryCount>=querySize){

homeLocationsFetchFinishedFlag = true;

if(locationsFetchFinishedFlag)

dataFetchFinishedUI();

else if(localDbEmptyFlag)

localDbEmptyUI();

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

internetDisconncetedUI();

Log.d(LogTags.MatchFound\_TAG, "onCancelled: home location query failed "+databaseError.getMessage());

} });

}

}

private void findMatchedLocations() {

localDbEmptyFlag = false;

locationsFetchFinishedFlag = false;

matchedLocationPosition = 0;

locationQueryCount = 0;

if(internetAvailable) {

retryButton.setVisibility(View.GONE);

retryButton.setEnabled(false);

}

matchedLocations.clear();

locationListAdapter = new LocationListAdapter(this, matchedLocations);

locationRecyclerView.setAdapter(locationListAdapter);

roomDatabase.databaseWriteExecutor.execute(new Runnable() {

@Override

public void run() {

// fetch from local db and query firebase

retrievedDatas = visitedLocationsDao.fetchAll();

dataSize = retrievedDatas.size();

if(dataSize==0){

// local database empty

localDbEmptyFlag = true;

if(homeLocationsFetchFinishedFlag) {

runOnUiThread(new Runnable() {

@Override

public void run() {

localDbEmptyUI();

}

});

}

return;

}

for (VisitedLocations currentEntry: retrievedDatas)

{

// format = "latLon\_dateTime"

String[] splitter = currentEntry.splitPrimaryKey();

// firebase query values

final String key = currentEntry.getATencodedlatlon();

final String dateTime = splitter[1];

Log.d(LogTags.MatchFound\_TAG, "run: query key = "+key +" date time = "+dateTime);

if(!Internet.isInternetAvailable(getApplicationContext())){

runOnUiThread(new Runnable() {

@Override

public void run() {

internetDisconncetedUI();

}

});

return;

}

// query in firebase

firebaseReference = FirebaseDatabase.getInstance().getReference().child("infectedLocations").child(key).child(dateTime);

firebaseReference.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

if(dataSnapshot.getValue()!=null){

// INFECTED LOCATION MATCH FOUND!

String latLon = key;

long verifiedCount = (long) dataSnapshot.child("verifiedCount").getValue();

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

MatchedLocation matchedLocation = new MatchedLocation(latLon, dateTime, verifiedCount, unverifiedCount);

matchedLocations.add(matchedLocation);

locationListAdapter.notifyItemInserted(matchedLocationPosition);

// start address fetch service

addressReceiver.startAddressFetchService(

ShowMatchedLocationsActivity.this,

matchedLocation.getBlLatitude(),

matchedLocation.getBlLongitude(),

matchedLocationPosition

);

matchedLocationPosition++;

}

locationQueryCount++;

if(locationQueryCount>=dataSize){

if(matchedLocations.isEmpty()){

// no locations match

locationsFetchFinishedFlag = true;

if(matchedHomeLocations.isEmpty()) {

// no home locations match either

// show no match found

runOnUiThread(new Runnable() {

@Override

public void run() {

noMatchFoundUI();

}

});

}

else {

// no location match

// but home location matched show finish UI

runOnUiThread(new Runnable() {

@Override

public void run() {

dataFetchFinishedUI();

}

});

}

}

}

}

@Override

public void onCancelled(@NonNull DatabaseError databaseError) {

// internet connection lost

runOnUiThread(new Runnable() {

@Override

public void run() {

internetDisconncetedUI();

}

});

}

});

}

}

});

}

private void internetDisconncetedUI() {

internetAvailable = false;

progressBar.setVisibility(View.INVISIBLE);

//linearLayout.setVisibility(View.INVISIBLE);

progressBarText.setText(getText(R.string.internet\_disconnected\_text));

progressBarText.setVisibility(View.VISIBLE);

retryButton.setEnabled(true);

retryButton.setVisibility(View.VISIBLE);

Log.d("removethis", "internetDisconncetedUI: visible");

Toast.makeText(this, getText(R.string.no\_internet\_toast), Toast.LENGTH\_LONG)

.show();

}

private void dataFetchFinishedUI(){

retryButton.setEnabled(false);

progressBarText.setVisibility(View.GONE);

progressBar.setVisibility(View.GONE);

if(internetAvailable) {

retryButton.setVisibility(View.GONE);

retryButton.setEnabled(false);

}

Toast.makeText(this, getText(R.string.finished\_progressbar\_text), Toast.LENGTH\_LONG)

.show();

}

private void noMatchFoundUI(){

progressBar.setVisibility(View.INVISIBLE);

if(internetAvailable) {

retryButton.setVisibility(View.GONE);

retryButton.setEnabled(false);

}

progressBarText.setVisibility(View.VISIBLE);

progressBarText.setText(getText(R.string.no\_match\_found\_text));

}

private void localDbEmptyUI(){

progressBar.setVisibility(View.INVISIBLE);

//linearLayout.setVisibility(View.INVISIBLE);

if(internetAvailable) {

retryButton.setVisibility(View.GONE);

retryButton.setEnabled(false);

}

progressBarText.setVisibility(View.VISIBLE);

progressBarText.setText(getText(R.string.local\_db\_empty\_text));

}

public void retryClicked(View view) {

internetAvailable = true;

progressBar.setVisibility(View.VISIBLE);

progressBarText.setVisibility(View.VISIBLE);

progressBarText.setText(getText(R.string.loading\_progressbar\_text));

findHomeMatchedLocations();

findMatchedLocations(); }

private int updateCount = 0;

@Override

public void updateAddress(String address, int listPosition) {

/\*

address received here

\*/

matchedLocations.get(listPosition).setAddress(address);

locationListAdapter.notifyItemChanged(listPosition);

Log.d(LogTags.MatchFound\_TAG, "updateAddress: address = "+matchedLocations.get(listPosition).toString());

updateCount++;

if(updateCount>=matchedLocations.size()){

locationsFetchFinishedFlag = true;

updateCount = 0;

if(homeLocationsFetchFinishedFlag)

dataFetchFinishedUI();

}}