/\*\*

\* Copyright 2019 Google LLC.

\* SPDX-License-Identifier: Apache-2.0

\*/

function getEnvironment() {

 var environment = {

   spreadsheetID: "1VeagIfZ5ZpAuOtCf6M2knTCNrSm1m9BlNkAJ5TCbSuY",

   firebaseUrl: "https://drsystem-139f0-default-rtdb.asia-southeast1.firebasedatabase.app/"

 };

 return environment;

}

// Creates a Google Sheets on change trigger for the specific sheet

function createSpreadsheetEditTrigger(sheetID) {

 var triggers = ScriptApp.getProjectTriggers();

 var triggerExists = false;

 for (var i = 0; i < triggers.length; i++) {

   if (triggers[i].getTriggerSourceId() == sheetID) {

     triggerExists = true;

     break;

   }

 }

 if (!triggerExists) {

   var spreadsheet = SpreadsheetApp.openById(sheetID);

   ScriptApp.newTrigger("importSheet")

     .forSpreadsheet(spreadsheet)

     .onChange()

     .create();

 }

}

// Creates a form submission trigger for the specific form

function createFormSubmissionTrigger(formID) {

 var triggers = ScriptApp.getProjectTriggers();

 var triggerExists = false;

 for (var i = 0; i < triggers.length; i++) {

   if (triggers[i].getTriggerSourceId() == formID) {

     triggerExists = true;

     break;

   }

 }

 if (!triggerExists) {

   ScriptApp.newTrigger("importSheet")

     .forForm(formID)

     .onFormSubmit()

     .create();

 }

}

// Delete all the existing triggers for the project

function deleteTriggers() {

 var triggers = ScriptApp.getProjectTriggers();

 for (var i = 0; i < triggers.length; i++) {

   ScriptApp.deleteTrigger(triggers[i]);

 }

}

// Initialize

function initialize(e) {

 writeDataToFirebase(getEnvironment().spreadsheetID);

}

// Write the data to the Firebase URL

function writeDataToFirebase(sheetID) {

 var ss = SpreadsheetApp.openById(sheetID);

 SpreadsheetApp.setActiveSpreadsheet(ss);

 createSpreadsheetEditTrigger(sheetID);

 var sheets = ss.getSheets();

 for (var i = 0; i < sheets.length; i++) {

   importSheet(sheets[i]);

   SpreadsheetApp.setActiveSheet(sheets[i]);

 }

}

// A utility function to generate nested object when

// given a keys in array format

function assign(obj, keyPath, value) {

 lastKeyIndex = keyPath.length - 1;

 for (var i = 0; i < lastKeyIndex; ++i) {

   key = keyPath[i];

   if (!(key in obj)) obj[key] = {};

   obj = obj[key];

 }

 obj[keyPath[lastKeyIndex]] = value;

}

// Import each sheet when there is a change

function importSheet() {

 var sheet = SpreadsheetApp.getActiveSheet();

 var name = sheet.getName();

 var data = sheet.getDataRange().getValues();

 var dataToImport = {};

 for (var i = 1; i < data.length; i++) {

   dataToImport[data[i][0]] = {};

   for (var j = 0; j < data[0].length; j++) {

     assign(dataToImport[data[i][0]], data[0][j].split("\_\_"), data[i][j]);

   }

 }

 var token = ScriptApp.getOAuthToken();

 var firebaseUrl =

   getEnvironment().firebaseUrl + sheet.getParent().getId() + "/" + name;

 var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, token);

 base.setData("", dataToImport);

}

// Set up the form submission trigger

createFormSubmissionTrigger("17UM1o7NEglg6Lioe2ktCgrG7\_vNTtIEoA\_72Dt25Vkw");

