/\*\*

\* CTĐ, CTCT – API cho PWA

\* Hỗ trợ Google Sheets + Excel (.xlsx)

\*

\* A. Lấy câu hỏi:

\* GET ?action=questions&sheetId=...&tab=Câu%20hỏi

\* GET ?action=questionsXlsx&fileId=...&tab=Câu%20hỏi

\* GET ?action=mixQuestions&folderId=...&limit=30&perFile=0&tab=Câu%20hỏi

\* B. Danh sách ngân hàng:

\* GET ?action=listBanks&folderId=...

\* C. Tài liệu PDF:

\* GET ?action=materials&folderId=...

\* D. Ghi điểm:

\* POST JSON { examCode,name,unit,position,score,total,details[] }

\* E. Tra cứu:

\* GET ?action=myResults&name=...&unit=...

\* GET ?action=leaderboard&group=unit|person&period=this\_week|last\_week|this\_month|last\_month|7d

\*

\* LƯU Ý: Dùng Drive.Files -> bật Advanced Google Services (Drive API).

\*/

const TZ = 'Asia/Ho\_Chi\_Minh';

const SHEET\_QUESTIONS = 'Câu hỏi';

/\* ==== Cấu trúc bảng điểm ==== \*/

const SUM\_SHEET\_NAME = 'Tổng hợp';

const DET\_SHEET\_NAME = 'Chi tiết';

const SUM\_HEADERS = ['Thời gian','Mã đề','Họ và tên','Đơn vị','Chức vụ','Điểm','Tổng số câu'];

const DET\_HEADERS = ['Thời gian','Mã đề','Họ và tên','Đơn vị','Chức vụ','Số thứ tự','Câu hỏi','Đáp án đã chọn','Đáp án đúng','Kết quả'];

/\* ==== Tiện ích ==== \*/

function nowVN(){ return Utilities.formatDate(new Date(), TZ, 'dd/MM/yyyy HH:mm:ss'); }

function json\_(o){ return ContentService.createTextOutput(JSON.stringify(o)).setMimeType(ContentService.MimeType.JSON); }

function text\_(s){ return ContentService.createTextOutput(String(s)).setMimeType(ContentService.MimeType.TEXT); }

function getOrCreateSheet(ss, name){ return ss.getSheetByName(name) || ss.insertSheet(name); }

function ensureHeaders(sh, headers){

const rng = sh.getRange(1,1,1,headers.length);

const cur = rng.getValues()[0];

if (cur.join('').trim()==='' || cur.length!==headers.length){ rng.setValues([headers]); return; }

let diff=false; for (let i=0;i<headers.length;i++) if (cur[i]!==headers[i]){diff=true;break;}

if (diff) rng.setValues([headers]);

}

function headerIndexMap\_(headers){ const m={}; headers.forEach((h,i)=>m[String(h).trim()]=i); return m; }

function norm\_(s){

s = String(s||'').trim().toLowerCase();

const map = { á:'a',à:'a',ạ:'a',ả:'a',ã:'a',ă:'a',ắ:'a',ằ:'a',ặ:'a',ẳ:'a',ẵ:'a',â:'a',ấ:'a',ầ:'a',ậ:'a',ẩ:'a',ẫ:'a',

é:'e',è:'e',ẹ:'e',ẻ:'e',ẽ:'e',ê:'e',ế:'e',ề:'e',ệ:'e',ể:'e',ễ:'e', í:'i',ì:'i',ị:'i',ỉ:'i',ĩ:'i',

ó:'o',ò:'o',ọ:'o',ỏ:'o',õ:'o',ô:'o',ố:'o',ồ:'o',ộ:'o',ổ:'o',ỗ:'o',ơ:'o',ớ:'o',ờ:'o',ợ:'o',ở:'o',ỡ:'o',

ú:'u',ù:'u',ụ:'u',ủ:'u',ũ:'u',ư:'u',ứ:'u',ừ:'u',ự:'u',ử:'u',ữ:'u', ý:'y',ỳ:'y',ỵ:'y',ỷ:'y',ỹ:'y', đ:'d' };

return s.replace(/[áàạảãăắằặẳẵâấầậẩẫéèẹẻẽêếềệểễíìịỉĩóòọỏõôốồộổỗơớờợởỡúùụủũưứừựửữýỳỵỷỹđ]/g,c=>map[c]||c);

}

function shuffle\_(a){ for(let i=a.length-1;i>0;i--){ const j=Math.floor(Math.random()\*(i+1)); [a[i],a[j]]=[a[j],a[i]]; } return a; }

/\* tìm cột theo nhiều nhãn khác nhau (phòng khi tiêu đề thay đổi) \*/

function findColIndex\_(header, keys){

const low = header.map(h=>String(h||'').toLowerCase().trim());

for (const k of keys){

const i = low.indexOf(String(k).toLowerCase().trim());

if (i >= 0) return i;

}

return -1;

}

/\* parse thời gian trong sheet “Tổng hợp” (hh:mm dd/MM/yyyy hoặc dd/MM/yyyy …) \*/

function parseVNDate\_(v){

if (v instanceof Date) return v;

const s = String(v||'').trim();

// dd/MM/yyyy HH:mm:ss or dd/MM/yyyy HH:mm or dd/MM/yyyy

const m = s.match(/(\d{1,2})\/(\d{1,2})\/(\d{4})(?:\s+(\d{1,2}):(\d{1,2})(?::(\d{1,2}))?)?/);

if (m){

const d = parseInt(m[1],10), M=parseInt(m[2],10)-1, y=parseInt(m[3],10);

const hh=parseInt(m[4]||'0',10), mm=parseInt(m[5]||'0',10), ss=parseInt(m[6]||'0',10);

return new Date(y,M,d,hh,mm,ss);

}

return null;

}

/\* ==== Đọc câu hỏi từ 1 Spreadsheet (tab “Câu hỏi”) ==== \*/

function readQuestionsFromSpreadsheet\_(ss, tabName){

const sh = ss.getSheetByName(tabName) || ss.getSheets()[0];

const values = sh.getDataRange().getValues();

let start = 1;

const h = values[0].map(v=>String(v).toLowerCase());

const looksHeader = h.join(' ').includes('câu') || h.join(' ').includes('đáp') || h.join(' ').includes('question');

if (!looksHeader) start = 0;

const out = [];

for (let r=start; r<values.length; r++){

const [q,A,B,C,D,correct,explanation] = values[r]; // explanation nếu có

if (!q) continue;

const ans = String(correct||'').trim().toUpperCase();

if (!['A','B','C','D'].includes(ans)) continue;

out.push({

question: String(q),

options: { A:String(A||''), B:String(B||''), C:String(C||''), D:String(D||'') },

answer: ans,

explanation: String(explanation||'')

});

}

return out;

}

/\* ==== Mã đề 4 chữ số, tránh trùng ==== \*/

function genExamCode4\_(){

const ss = SpreadsheetApp.getActiveSpreadsheet();

const det = ss.getSheetByName(DET\_SHEET\_NAME) || ss.insertSheet(DET\_SHEET\_NAME);

for (let tries=0; tries<20; tries++){

const code=('0000'+Math.floor(Math.random()\*10000)).slice(-4);

const hit = det.createTextFinder(code).matchEntireCell(true).findNext();

if (!hit) return code;

}

return Utilities.formatDate(new Date(), TZ, 'HHmm');

}

/\* ==== GHI KẾT QUẢ + TÔ MÀU Ô "Kết quả" (ĐÚNG/SAI) ==== \*/

function appendResultBlock\_(sum, det, ts, code, data) {

// 1) Ghi sheet "Tổng hợp"

sum.appendRow([

ts, code,

data.name || '', data.unit || '', data.position || '',

data.score || 0, data.total || 0

]);

// 2) Ghi sheet "Chi tiết"

const details = Array.isArray(data.details) ? data.details : [];

if (!details.length) return;

// Mỗi dòng chỉ điền đủ thông tin ở dòng đầu block

const rows = details.map((d, i) => [

i === 0 ? ts : '',

i === 0 ? code : '',

i === 0 ? (data.name || '') : '',

i === 0 ? (data.unit || '') : '',

i === 0 ? (data.position || '') : '',

d.index || '',

d.question || '',

d.chosen || '',

d.correct || '',

// chữ "ĐÚNG" / "SAI" vào cột "Kết quả"

d.isCorrect === true ? 'ĐÚNG' : 'SAI'

]);

const startRow = det.getLastRow() + 1;

det.getRange(startRow, 1, rows.length, rows[0].length).setValues(rows);

// 3) Tự wrap phần "Câu hỏi" để hiển thị hết nội dung

try {

const headers = det.getRange(1, 1, 1, det.getLastColumn()).getValues()[0];

const colQ = headers.indexOf('Câu hỏi') + 1;

if (colQ > 0) {

const n = Math.max(0, det.getLastRow() - 1);

if (n > 0) {

det.getRange(2, colQ, n, 1).setWrap(true).setVerticalAlignment('top');

det.autoResizeColumn(colQ);

}

}

} catch (\_) {}

// 4) Tô màu cột "Kết quả" theo yêu cầu

try {

const headers = det.getRange(1, 1, 1, det.getLastColumn()).getValues()[0];

const colRes = headers.indexOf('Kết quả') + 1;

if (colRes > 0) {

const GREEN = '#22C55E'; // xanh lá

const RED = '#DC2626'; // đỏ

const GOLD = '#FFD700'; // vàng

const bgs = [], fgs = [], fws = [], aligns = [];

for (const d of details) {

const ok = d.isCorrect === true;

bgs.push([ ok ? GREEN : RED ]);

fgs.push([ GOLD ]);

fws.push([ 'bold' ]);

aligns.push([ 'center' ]);

}

const rg = det.getRange(startRow, colRes, details.length, 1);

rg.setBackgrounds(bgs)

.setFontColors(fgs)

.setFontWeights(fws)

.setHorizontalAlignments(aligns);

}

} catch (\_) {}

}

/\* ====================== GET ====================== \*/

function doGet(e){

const p = e && e.parameter || {};

const action = (p.action||'').trim();

// Danh sách bộ đề trong 1 thư mục Drive (Sheets + XLSX)

if (action==='listBanks'){

const folderId=(p.folderId||'').trim();

if (!folderId) return json\_({banks:[], error:'missing\_folderId'});

const folder = DriveApp.getFolderById(folderId);

const banks=[];

const itSheets = folder.getFilesByType(MimeType.GOOGLE\_SHEETS);

while (itSheets.hasNext()){

const f=itSheets.next(); banks.push({id:f.getId(), title:f.getName(), type:'gsheet'});

}

const itXlsx = folder.getFilesByType('application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');

while (itXlsx.hasNext()){

const f=itXlsx.next(); banks.push({id:f.getId(), title:f.getName(), type:'xlsx'});

}

banks.sort((a,b)=>a.title.localeCompare(b.title,'vi'));

return json\_({banks});

}

// Câu hỏi từ 1 Google Sheet

if (action==='questions'){

const sheetId=(p.sheetId||'').trim();

const tab=(p.tab||SHEET\_QUESTIONS).trim();

const ss = sheetId ? SpreadsheetApp.openById(sheetId) : SpreadsheetApp.getActiveSpreadsheet();

return json\_({ questions: readQuestionsFromSpreadsheet\_(ss, tab) });

}

// Câu hỏi từ 1 Excel (.xlsx) -> copy tạm sang Google Sheets để đọc

if (action==='questionsXlsx'){

const fileId=(p.fileId||'').trim();

const tab=(p.tab||SHEET\_QUESTIONS).trim();

if (!fileId) return json\_({questions:[], error:'missing\_fileId'});

const tmp = Drive.Files.copy({title:'tmp\_'+fileId, mimeType:MimeType.GOOGLE\_SHEETS}, fileId);

try{

const ss=SpreadsheetApp.openById(tmp.id);

return json\_({ questions: readQuestionsFromSpreadsheet\_(ss, tab) });

} finally { try{ Drive.Files.trash(tmp.id);}catch(\_){ } }

}

// TRỘN NGẪU NHIÊN từ TOÀN THƯ MỤC (Sheets + XLSX)

if (action==='mixQuestions'){

const folderId=(p.folderId||'').trim();

const limit = Math.max(1, parseInt(p.limit||'30',10));

const perFile = Math.max(0, parseInt(p.perFile||'0',10)); // 0 = không hạn chế

const tab=(p.tab||SHEET\_QUESTIONS).trim();

if (!folderId) return json\_({questions:[], error:'missing\_folderId'});

const folder = DriveApp.getFolderById(folderId);

const pool=[];

function pushSome(src){

if (!src || !src.length) return;

shuffle\_(src);

const take = perFile>0? Math.min(perFile, src.length) : src.length;

for (let i=0;i<take;i++) pool.push(src[i]);

}

// Sheets

const itSheets2=folder.getFilesByType(MimeType.GOOGLE\_SHEETS);

while (itSheets2.hasNext()){

const f=itSheets2.next();

try{ pushSome(readQuestionsFromSpreadsheet\_(SpreadsheetApp.openById(f.getId()), tab)); }catch(\_){}

}

// XLSX

const itXlsx2=folder.getFilesByType('application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');

while (itXlsx2.hasNext()){

const f=itXlsx2.next();

try{

const tmp = Drive.Files.copy({title:'tmp\_'+f.getId(), mimeType:MimeType.GOOGLE\_SHEETS}, f.getId());

try{ pushSome(readQuestionsFromSpreadsheet\_(SpreadsheetApp.openById(tmp.id), tab)); }

finally{ try{ Drive.Files.trash(tmp.id);}catch(\_){ } }

}catch(\_){}

}

shuffle\_(pool);

return json\_({ questions: pool.slice(0, Math.min(limit, pool.length)), totalPool: pool.length });

}

// Tài liệu PDF

if (action==='materials'){

const folderId=(p.folderId||'').trim();

if (!folderId) return json\_({files:[], error:'missing\_folderId'});

const folder=DriveApp.getFolderById(folderId);

const it=folder.getFiles();

const files=[];

while (it.hasNext()){

const f=it.next(); if (f.getMimeType()!==MimeType.PDF) continue;

files.push({ id:f.getId(), title:f.getName(),

url:`https://drive.google.com/file/d/${f.getId()}/preview`, updated:f.getLastUpdated() });

}

files.sort((a,b)=> new Date(b.updated)-new Date(a.updated));

return json\_({files});

}

// Tra cứu: kết quả cá nhân

if (action==='myResults'){

const name=(p.name||'').trim(), unit=(p.unit||'').trim();

if (!name || !unit) return json\_({rows:[]});

const ss=SpreadsheetApp.getActiveSpreadsheet();

const sum=getOrCreateSheet(ss, SUM\_SHEET\_NAME);

const data=sum.getDataRange().getValues(); if (data.length<2) return json\_({rows:[]});

const H = headerIndexMap\_(data[0]);

const iTime=H['Thời gian'], iCode=H['Mã đề'], iName=H['Họ và tên'], iUnit=H['Đơn vị'], iScore=H['Điểm'], iTotal=H['Tổng số câu'];

const nameQ=norm\_(name), unitQ=norm\_(unit);

const out=[];

for (let r=1;r<data.length;r++){

const row=data[r];

if (norm\_(row[iName])===nameQ && norm\_(row[iUnit])===unitQ){

out.push({ time:row[iTime], examCode:row[iCode], score:row[iScore], total:row[iTotal] });

}

}

try{ out.sort((a,b)=> new Date(b.time)-new Date(a.time)); }catch(\_){}

return json\_({ rows: out });

}

// Bảng xếp hạng: theo đơn vị hoặc theo cá nhân + lọc thời gian

if (action === 'leaderboard'){

try{

const group = (p.group || 'unit').trim(); // 'unit' | 'person'

const period = (p.period || '').trim(); // this\_week | last\_week | this\_month | last\_month | 7d | '' (all)

const fromStr = (p.from || '').trim();

const toStr = (p.to || '').trim();

function startOfWeek(d){ const x=new Date(d); const dow=(x.getDay()+6)%7; x.setDate(x.getDate()-dow); x.setHours(0,0,0,0); return x; }

function endOfWeek(d){ const s=startOfWeek(d); const e=new Date(s); e.setDate(s.getDate()+7); return e; } // exclusive

function startOfMonth(d){ return new Date(d.getFullYear(), d.getMonth(), 1); }

function endOfMonth(d){ return new Date(d.getFullYear(), d.getMonth()+1, 1); } // exclusive

const today = new Date();

let from = null, to = null; // exclusive to

if (fromStr || toStr){

from = fromStr ? new Date(fromStr+'T00:00:00') : null;

to = toStr ? new Date(toStr +'T00:00:00') : null;

if (to) to.setDate(to.getDate()+1);

} else {

if (period === 'this\_week'){ from = startOfWeek(today); to = endOfWeek(today); }

else if (period === 'last\_week'){ const t=new Date(today); t.setDate(t.getDate()-7); from = startOfWeek(t); to = endOfWeek(t); }

else if (period === '7d'){ const t=new Date(today); t.setDate(t.getDate()-6); from = new Date(t.getFullYear(), t.getMonth(), t.getDate()); to = new Date(today.getFullYear(), today.getMonth(), today.getDate()+1); }

else if (period === 'this\_month'){ from = startOfMonth(today); to = endOfMonth(today); }

else if (period === 'last\_month'){ const t=new Date(today.getFullYear(), today.getMonth()-1, 15); from = startOfMonth(t); to = endOfMonth(t); }

// else all time

}

const ss=SpreadsheetApp.getActiveSpreadsheet();

const sum=getOrCreateSheet(ss, SUM\_SHEET\_NAME);

const data=sum.getDataRange().getValues();

if (data.length<2) return json\_({ rows: [], group, reason:'no\_rows' });

const header = data[0];

const iTime = findColIndex\_(header, ['thời gian','thoi gian','time','date']);

const iName = findColIndex\_(header, ['họ và tên','ho va ten','ten','name']);

const iUnit = findColIndex\_(header, ['đơn vị','don vi','unit']);

const iScore = findColIndex\_(header, ['điểm','diem','score']);

const iTotal = findColIndex\_(header, ['tổng số câu','tong so cau','tong so','total']);

if ([iTime,iUnit,iScore,iTotal].some(i => i < 0) || (group==='person' && iName<0)){

return json\_({ rows: [], group, reason:'missing\_columns', header });

}

const agg = {};

for (let r=1; r<data.length; r++){

const row = data[r];

const when = parseVNDate\_(row[iTime]);

if (from && when && when < from) continue;

if (to && when && when >= to) continue;

const unit = String(row[iUnit]||'').trim(); if (!unit) continue;

const score = Number(row[iScore]||0);

const total = Number(row[iTotal]||0);

const pct = total ? (score/total\*10) : 0;

let key, name;

if (group==='person'){

name = String(row[iName]||'').trim();

if (!name) continue;

key = name + '|' + unit;

} else {

key = unit;

}

if (!agg[key]) agg[key] = { count:0, sumPct:0, max:0, unit, name: name||'' };

agg[key].count++;

agg[key].sumPct += pct;

if (score > agg[key].max) agg[key].max = score;

}

const rows = Object.keys(agg).map(k=>{

const a = agg[k];

const avg = a.count ? (a.sumPct / a.count) : 0;

return (group==='person')

? { name:a.name, unit:a.unit, count:a.count, avg:avg.toFixed(2), max:a.max }

: { unit:a.unit, count:a.count, avg:avg.toFixed(2), max:a.max };

});

rows.sort((x,y)=> (Number(y.avg)-Number(x.avg)) || (Number(y.max)-Number(x.max)) || (Number(y.count)-Number(x.count)));

return json\_({ group, rows, period, from, to });

}catch(err){

return json\_({ group:(p.group||'unit'), rows:[], error:String(err) });

}

}

// Ping

return text\_('UP');

}

/\* ====================== POST ====================== \*/

function doPost(e){

const lock = LockService.getDocumentLock(); lock.waitLock(30000);

try{

const ss = SpreadsheetApp.getActiveSpreadsheet();

const sum = getOrCreateSheet(ss, SUM\_SHEET\_NAME);

const det = getOrCreateSheet(ss, DET\_SHEET\_NAME);

ensureHeaders(sum, SUM\_HEADERS);

ensureHeaders(det, DET\_HEADERS);

const data = JSON.parse(e.postData.contents || '{}');

// Nếu mã đề thiếu/không hợp lệ/đã tồn tại -> tự sinh

let code = (data.examCode||'').toString().trim();

const existed = code && det.createTextFinder(code).matchEntireCell(true).findNext();

if (!/^\d{4}$/.test(code) || existed) code = genExamCode4\_();

const ts = nowVN();

appendResultBlock\_(sum, det, ts, code, data);

return text\_('OK');

} finally { lock.releaseLock(); }

}