// Configuration

const CONFIG = {

  SESSIONS: [

    { name: "Main Session", start: "09:15", end: "13:30" }

  ],

  BREAK\_DURATION: 30 // minutes for extended break threshold

};

function doGet(e) {

  return handleAttendance(e);

}

function doPost(e) {

  return handleAttendance(e);

}

function handleAttendance(e) {

  // Validate UID

  if (!e.parameter.uid || e.parameter.uid.trim() === "") {

    return ContentService.createTextOutput("Error: UID is required");

  }

  var ss = SpreadsheetApp.getActiveSpreadsheet();

  var studentsSheet = ss.getSheetByName("Students");

  if (studentsSheet == null) {

    return ContentService.createTextOutput("Error: Students sheet not found");

  }

  var uid = e.parameter.uid.trim();

  var now = new Date();

  var date = Utilities.formatDate(now, "GMT+5:30", "dd-MMM-yyyy");

  var time = Utilities.formatDate(now, "GMT+5:30", "HH:mm:ss");

  // Check if student exists - optimized for performance

  var studentData = studentsSheet.getDataRange().getValues();

  var studentMap = {};

  for (var j = 1; j < studentData.length; j++) {

    studentMap[studentData[j][1]] = studentData[j][0];

  }

  var studentName = studentMap[uid];

  if (!studentName) {

    return ContentService.createTextOutput("Error: Unauthorized UID - " + uid);

  }

  // Create or get today's sheet

  var sheet = ss.getSheetByName(date);

  if (sheet == null) {

    sheet = ss.insertSheet(date);

    sheet.appendRow(["Student Name", "UID", "Date", "Time In", "Time Out"]);

  }

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

  // Find the last entry for this UID today

  var lastEntry = null;

  var lastEntryRow = 0;

  for (var i = data.length - 1; i >= 1; i--) {

    if (data[i][1] == uid) {

      lastEntry = data[i];

      lastEntryRow = i + 1;

      break;

    }

  }

  if (lastEntry) {

    if (lastEntry[4] == "" || lastEntry[4] == null) {

      // Update Time Out

      sheet.getRange(lastEntryRow, 5).setValue(time);

      return ContentService.createTextOutput("Time Out updated: " + time);

    } else {

      // Add new Time In

      sheet.appendRow([studentName, uid, date, time, ""]);

      return ContentService.createTextOutput("New Time In marked: " + time);

    }

  } else {

    // First entry of the day

    sheet.appendRow([studentName, uid, date, time, ""]);

    return ContentService.createTextOutput("First Time In marked: " + time);

  }

}

// Batch processing for multiple UIDs

function batchUpdateAttendance(uidList) {

  var results = [];

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

    try {

      var result = handleAttendance({parameter: {uid: uidList[i]}});

      results.push({

        uid: uidList[i],

        status: result.getContent(),

        success: true

      });

    } catch (e) {

      results.push({

        uid: uidList[i],

        status: "Error: " + e.toString(),

        success: false

      });

    }

  }

  return ContentService.createTextOutput(JSON.stringify(results))

    .setMimeType(ContentService.MimeType.JSON);

}

// Function to check who's currently present

function getCurrentlyPresent() {

  var ss = SpreadsheetApp.getActiveSpreadsheet();

  var date = Utilities.formatDate(new Date(), "GMT+5:30", "dd-MMM-yyyy");

  var sheet = ss.getSheetByName(date);

  if (!sheet) return [];

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

  var presentStudents = [];

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

    if (data[i][3] && !data[i][4]) { // Has Time In but no Time Out

      presentStudents.push({

        name: data[i][0],

        uid: data[i][1],

        timeIn: data[i][3]

      });

    }

  }

  return presentStudents;

}

// Function to generate 4:30 PM summary on the same sheet

function generate430PMSummary() {

  var ss = SpreadsheetApp.getActiveSpreadsheet();

  var date = Utilities.formatDate(new Date(), "GMT+5:30", "dd-MMM-yyyy");

  var sheet = ss.getSheetByName(date);

  var studentsSheet = ss.getSheetByName("Students");

  if (!studentsSheet) {

    console.log("Students sheet not found");

    return;

  }

  // Get total number of students

  var totalStudents = studentsSheet.getLastRow() - 1; // Minus header row

  if (!sheet) {

    console.log("No attendance data for today");

    return;

  }

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

  // Process attendance data

  var attendedStudents = {};

  var notTappedOut = [];

  var studentBreaks = {}; // Track breaks per student

  // First pass - identify all students and their entries

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

    var uid = data[i][1];

    if (!attendedStudents[uid]) {

      attendedStudents[uid] = {

        name: data[i][0],

        entries: []

      };

    }

    attendedStudents[uid].entries.push({

      timeIn: data[i][3],

      timeOut: data[i][4]

    });

  }

  // Second pass - analyze breaks and current status

  for (var uid in attendedStudents) {

    var student = attendedStudents[uid];

    var extendedBreaks = 0;

    // Check each entry

    for (var j = 0; j < student.entries.length; j++) {

      var entry = student.entries[j];

      // Check if currently in (last entry has no tap out)

      if (j === student.entries.length - 1 && entry.timeIn && !entry.timeOut) {

        notTappedOut.push({

          name: student.name,

          uid: uid,

          timeIn: entry.timeIn

        });

      }

      // Check for extended breaks (if there's a next entry)

      if (j < student.entries.length - 1 && entry.timeOut) {

        var breakStart = new Date(date + " " + entry.timeOut);

        var breakEnd = new Date(date + " " + student.entries[j + 1].timeIn);

        var breakDuration = (breakEnd - breakStart) / (1000 \* 60); // in minutes

        if (breakDuration >= CONFIG.BREAK\_DURATION) {

          extendedBreaks++;

        }

      }

    }

    if (extendedBreaks > 0) {

      studentBreaks[uid] = {

        name: student.name,

        count: extendedBreaks

      };

    }

  }

  var presentCount = Object.keys(attendedStudents).length;

  var absentCount = totalStudents - presentCount;

  // Add summary to the sheet after some blank rows

  var lastRow = sheet.getLastRow();

  var summaryStartRow = lastRow + 3; // Leave 2 blank rows

  // Add summary header

  sheet.getRange(summaryStartRow, 1).setValue("DAILY ATTENDANCE SUMMARY").setFontWeight("bold").setFontSize(14);

  summaryStartRow++;

  // Summary data

  var summaryData = [

    ["Date:", date],

    ["Session Time:", CONFIG.SESSIONS[0].start + " - " + CONFIG.SESSIONS[0].end],

    ["Report Generated:", Utilities.formatDate(new Date(), "GMT+5:30", "HH:mm:ss")],

    [""],

    ["Total Students:", totalStudents],

    ["Present:", presentCount],

    ["Absent:", absentCount],

    ["Attendance %:", ((presentCount/totalStudents) \* 100).toFixed(2) + "%"],

    [""],

    ["Students with Extended Breaks (30+ mins):", Object.keys(studentBreaks).length],

    ["Students who haven't tapped out:", notTappedOut.length]

  ];

  // Write summary data

  for (var k = 0; k < summaryData.length; k++) {

    if (summaryData[k][0] !== "") {

      sheet.getRange(summaryStartRow + k, 1).setValue(summaryData[k][0]).setFontWeight("bold");

      if (summaryData[k][1] !== undefined) {

        sheet.getRange(summaryStartRow + k, 2).setValue(summaryData[k][1]);

      }

    }

  }

  // Add details of students with extended breaks if any

  if (Object.keys(studentBreaks).length > 0) {

    summaryStartRow += summaryData.length + 1;

    sheet.getRange(summaryStartRow, 1).setValue("Students with Extended Breaks:").setFontWeight("bold");

    summaryStartRow++;

    for (var uid in studentBreaks) {

      sheet.getRange(summaryStartRow, 1).setValue(studentBreaks[uid].name);

      sheet.getRange(summaryStartRow, 2).setValue(uid);

      sheet.getRange(summaryStartRow, 3).setValue(studentBreaks[uid].count + " extended break(s)");

      summaryStartRow++;

    }

  }

  // Add details of students who haven't tapped out if any

  if (notTappedOut.length > 0) {

    summaryStartRow += 2;

    sheet.getRange(summaryStartRow, 1).setValue("Students who haven't tapped out:").setFontWeight("bold");

    summaryStartRow++;

    for (var m = 0; m < notTappedOut.length; m++) {

      sheet.getRange(summaryStartRow, 1).setValue(notTappedOut[m].name);

      sheet.getRange(summaryStartRow, 2).setValue(notTappedOut[m].uid);

      sheet.getRange(summaryStartRow, 3).setValue("In since: " + notTappedOut[m].timeIn);

      summaryStartRow++;

    }

  }

  Logger.log("4:30 PM Summary generated successfully");

}

// Set up daily trigger for 4:30 PM summary report

function setupDailyTrigger() {

  // First, set the script timezone to IST

  var ss = SpreadsheetApp.getActiveSpreadsheet();

  ss.setSpreadsheetTimeZone("Asia/Kolkata");

  // Delete any existing triggers for this function to avoid duplicates

  var triggers = ScriptApp.getProjectTriggers();

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

    if (triggers[i].getHandlerFunction() === 'generate430PMSummary') {

      ScriptApp.deleteTrigger(triggers[i]);

    }

  }

  // Create new trigger for 4:30 PM IST

  ScriptApp.newTrigger('generate430PMSummary')

    .timeBased()

    .everyDays(1)

    .atHour(16)

    .nearMinute(30)

    .create();

  Logger.log("Daily summary trigger set up successfully for 4:30 PM IST");

}

// Function to manually run setup (call this once from Script Editor)

function runSetup() {

  setupDailyTrigger();

}

// Manual function to test summary generation

function testSummary() {

  generate430PMSummary();

}

// Helper function to get attendance status for API

function getAttendanceStatus() {

  var currentlyPresent = getCurrentlyPresent();

  var ss = SpreadsheetApp.getActiveSpreadsheet();

  var date = Utilities.formatDate(new Date(), "GMT+5:30", "dd-MMM-yyyy");

  var studentsSheet = ss.getSheetByName("Students");

  var totalStudents = studentsSheet ? studentsSheet.getLastRow() - 1 : 0;

  return {

    date: date,

    totalStudents: totalStudents,

    currentlyPresent: currentlyPresent.length,

    presentStudents: currentlyPresent

  };

}