// Constants and Configuration

var TELEGRAM\_BOT\_TOKEN = "6479253293:AAFjCdf91FWkA5\_r162\_Kjgej2e3miqtx2g";

var SHEET\_ID = "101JN0hqXmQqM-\_RKKxJxHoWK31dCbwfHOikUy\_2Yhyk";

var USERS\_SHEET\_NAME = "UsersSheet";

var GUESTS\_SHEET\_NAME = "GuestsSheet";

var url = "https://api.telegram.org/bot" + TELEGRAM\_BOT\_TOKEN;

var webAppUrl = "https://script.google.com/macros/s/AKfycbzFlMq1Fx4oW1l\_BDN6EZvp8EsTLQ-isZi5ebbkCMHlGfJwDqKzk5-Tb7MupKJFn-0hQQ/exec";

//! STATES SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

const States = {

  Unknown: -1,

  Unregistered: 0,

  Registering\_FirstNameReq: 1,

  Registering\_LastNameReq: 2,

  Registering\_MobileReq: 3,

  Registering\_WaitForAccept: 4,

  Registered: 5,

  Registered\_Expired: 6,

  Registered\_InviteGuest\_NCReq: 7,

  Registered\_InviteGuest\_MobileReq: 8,

  Registered\_InviteGuest\_DateReq: 9,

  Registered\_InviteGuest\_HourReq: 10,

};

function GetUserState(userId) {

  if (isUserIdAbsentInUsersSheet(userId)) {

    return States.Unregistered;

  } else if (isUserIdExistInUsersSheet(userId)) {

    return States.Registering\_FirstNameReq;

  } else if (isUserFirstNameExist(userId)) {

    return States.Registering\_LastNameReq;

  } else if (isUserLastNameExist(userId)) {

    return States.Registering\_MobileReq;

  } else if (isUserMobileExist(userId)) {

    return States.Registering\_WaitForAccept;

  } else if (isUserContractNotExpired(userId)) {

    if (isUserIdExistInGuestsSheet(userId)) {

      return States.Registered\_InviteGuest\_NCReq;

    } else if (isGuestNationalCodeExist(userId)) {

      return States.Registered\_InviteGuest\_MobileReq;

    } else if (isGuestMobileExist(userId)) {

      return States.Registered\_InviteGuest\_DateReq;

    } else if (isGuestDateExist(userId)) {

      return States.Registered\_InviteGuest\_HourReq;

    }

    return States.Registered;

  } else if (isUserContractExpired(userId)) {

    return States.Registered\_Expired;

  }

  return States.Unknown;

}

//! WEB APP FUNCTIONS WEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPPWEBAPP

function setWebhook() {

  var response = UrlFetchApp.fetch(url + "/setWebhook?url=" + webAppUrl);

  Logger.log(response.getContentText());

}

function deleteWebhook() {

  try {

    var response = UrlFetchApp.fetch(url + "/deleteWebhook");

    Logger.log(response.getContentText());

  } catch (error) {

    Logger.log("Error in deleteWebhook: " + error);

  }

}

function sendMessage(chatId, text) {

  var response = UrlFetchApp.fetch(url + "/sendMessage?chat\_id=" + chatId + "&text=" + text);

  Logger.log(response.getContentText());

}

//! STORE FUNCTIONS +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

//? USER ######################################################################################################################################################################

function storeUserIdInUsersSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

  sheet.appendRow([new Date(), userId]);

}

function storeUserFirstName(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId) {

      // Check if all columns C, D, E, F are empty

      var isEmpty = true;

      for (var j = 2; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns C, D, E, F are empty, append information in the 3rd column (c)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 3); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "آیدی تلگرام کاربر در شیت یوزرها پیدا نشد."); }

    }

  }

}

function storeUserLastName(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the User first name exists in the 3rd column

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][2] !== "") {

      // Check if all columns D, E, F are empty

      var isEmpty = true;

      for (var j = 3; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns D, E, F are empty, append information in the 4th column (D)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 4); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "نام کاربر وارد نشده است"); }

    }

  }

}

function storeUserMobile(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the User last name exists in the 4rd column

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][3] !== "") {

      // Check if all columns E, F are empty

      var isEmpty = true;

      for (var j = 4; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns E, F are empty, append information in the 5th column (E)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 5); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "نام خانوادگی کاربر وارد نشده است"); }

    }

  }

}

//? GUEST #####################################################################################################################################################################

function storeUserIdInGuestsSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  // Check if any incomplete row contains the user ID

  var incompleteRowFound = false;

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

    var row = data[i];

    var isComplete = true;

    for (var j = 0; j < 7; j++) {

      if (row[j] === "") {

        isComplete = false;

        break;

      }

    }

    if (row[1] === userId && !isComplete) {

      incompleteRowFound = true;

      break;

    }

  }

  // If no incomplete row found, append a new row

  if (!incompleteRowFound) {

    sheet.appendRow([new Date(), userId]);

  }

}

function storeGuestNationalCode(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId) {

      // Check if all columns C, D, E, F, G are empty

      var isEmpty = true;

      for (var j = 3; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns C, D, E, F, G are empty, append information in the 3rd column (C)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 3); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "آیدی تلگرام کاربر در شیت مهمان ها پیدا نشد."); }

    }

  }

}

function storeGuestMobile(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the guest national code exists in the 3rd column

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][2] !== "") {

      // Check if all columns D, E, F, G are empty

      var isEmpty = true;

      for (var j = 3; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns D, E, F, G are empty, append information in the 4th column (D)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 4); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "کد ملی مهمان وارد نشده است."); }

    }

  }

}

function storeGuestDate(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the guest mobile exists in the 4th column

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][3] !== "") {

      // Check if all columns E, F, G are empty

      var isEmpty = true;

      for (var j = 4; j <= 5; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns E, F, G are empty, append information in the 5th column (E)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 5); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "شماره تلفن مهمان وارد نشده است."); }

    }

  }

}

function storeGuestHour(messageText) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the guest date exists in the 5th column

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][4] !== "") {

      // Check if all columns F, G are empty

      var isEmpty = true;

      for (var j = 5; j <= 6; j++) {

        if (data[i][j] !== "") {

          isEmpty = false;

          break;

        }

      } // If all columns F, G are empty, append information in the 6th column (F)

      if (isEmpty) {

        var range = sheet.getRange(i + 1, 6); // Adjusted index by 1 since Google Sheets is 1-indexed

        range.setValue(messageText);

        break; // Exit the loop once the last record is found

      } else { sendMessage(userId, "تاریخ حضور مهمان وارد نشده است."); }

    }

  }

}

function storeGuestISODate(guestISODate) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index since JavaScript is 0-indexed

  // Find the last record of userId where the guest hour exists in the 6th column and 7th column is empty

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][5] !== "" && data[i][6] === "") {

      var range = sheet.getRange(i + 1, 7); // Adjusted index by 1 since Google Sheets is 1-indexed

      range.setValue(guestISODate);

      break; // Exit the loop once the last record is found

    } else {

      sendMessage(userId, "ساعت ورود مهمان وارد نشده است.");

    }

  }

}

//! IS ????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????

//? USER ??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????

function isUserIdExistInUsersSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    // Check columns must be filled

    if (data[i][0] !== "" && data[i][1] === userId) {

      // Check columns must be empty

      for (var j = 2; j <= 5; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isUserIdAbsentInUsersSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    if (data[i][1] === userId) {

      return false; // If the user ID is found in any row

    }

  }

  return true; // If the loop completes without finding userId

}

function isUserFirstNameExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    // Check columns must be filled

    if (data[i][0] !== "" && data[i][1] === userId && data[i][2] !== "") {

      // Check columns must be empty

      for (var j = 3; j <= 5; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isUserLastNameExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    // Check columns must be filled

    if (data[i][0] !== "" && data[i][1] === userId && data[i][2] !== "" && data[i][3] !== "") {

      // Check columns must be empty

      for (var j = 4; j <= 5; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isUserMobileExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    // Check columns must be filled

    if (data[i][0] !== "" && data[i][1] === userId && data[i][2] !== "" && data[i][3] !== "" && data[i][4] !== "") {

      // Check columns must be empty

      if (data[i][5] !== "") {

        return false; // If any column is not empty, return false

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isUserContractNotExpired(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

  var currentDate = new Date();

  currentDate.setHours(0, 0, 0, 0); // Set time to midnight for comparison

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

    if (data[i][1] === userId) {

      var expirationDate = new Date(data[i][5]);

      expirationDate.setHours(0, 0, 0, 0); // Set time to midnight for comparison

      if (currentDate <= expirationDate) {

        return true; // contract is not expired.

      } else {

        return false; // contract is expired.

      }

    }

  }

}

function isUserContractExpired(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

  var currentDate = new Date();

  currentDate.setHours(0, 0, 0, 0); // Set time to midnight for comparison

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

    if (data[i][1] === userId) {

      var expirationDate = data[i][5];

      expirationDate.setHours(0, 0, 0, 0); // Set time to midnight for comparison

      if (expirationDate !== "") {

        return currentDate >= expirationDate;

      }

    }

  }

  return true;

}

//? GUEST ?????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????

function isUserIdExistInGuestsSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId) {

      // Check if columns 3 to 7 are empty

      for (var j = 2; j <= 6; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isUserIdAbsentInGuestsSheet(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

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

    if (data[i][1] === userId) {

      return false; // If the user ID is found in any row

    }

  }

  return true; // If the loop completes without finding userId

}

function isGuestNationalCodeExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 3rd column (C) is filled.

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][2] !== "") {

      // Check if columns 4 to 7 are empty

      for (var j = 3; j <= 6; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isGuestMobileExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 4th column (D) is filled.

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][3] !== "") {

      // Check if columns 5 to 7 are empty

      for (var j = 4; j <= 6; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isGuestDateExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 5th column (E) is filled.

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][4] !== "") {

      // Check if columns 6 to 7 are empty

      for (var j = 5; j <= 6; j++) {

        if (data[i][j] !== "") {

          return false; // If any column is not empty, return false

        }

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isGuestHourExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 6th column (F) is filled.

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][5] !== "") {

      // Check if column 7 is empty

      if (data[i][6] !== "") {

        return false; // If column 7 is not empty, return false

      }

      return true; // If all conditions are met, return true

    }

  }

  return false; // User ID not found or conditions not met

}

function isGuestISODateExist(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 7th column (G) is filled.

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][6] !== "") {

      return true;

    }

  }

  return false;

}

function isValidUserFirstName(Info) {

  var InfoPattern = /^[\u0600-\u06FF\s]+$/u;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

function isValidUserLastName(Info) {

  var InfoPattern = /^[\u0600-\u06FF\s]+$/u;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

function isValidUserMobile(Info) {

  var InfoPattern = /^09\d{9}$/;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

function isValidNationalCode(Info) {

  var InfoPattern = /^(\d{10})+$/;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

function isValidGuestMobile(Info) {

  var InfoPattern = /^09\d{9}$/;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

function isValidDate(Info) {

  var InfoPattern = /^(0?[1-9]|[12]\d|3[01])\/(0?[1-9]|1[0-2])\/(\d{4})$/;

  var match = Info.match(InfoPattern);

  if (match !== null) {

    var solarYear = Number(match[3]);

    var solarMonth = Number(match[2]);

    var solarDay = Number(match[1]);

    // Check if the month is between 1 and 12

    if (solarMonth < 1 || solarMonth > 12) {

      return false;

    }

    // Check if the day is between 1 and 31 (or less depending on the month)

    var maxDay = new Date(solarYear, solarMonth, 0).getDate();

    if (solarDay < 1 || solarDay > maxDay) {

      return false;

    }

    try {

      // Convert solar date to Gregorian date using the Converter class

      var gregorianDate = Converter.persianToGregorian(solarYear, solarMonth, solarDay);

      var gregorianYear = gregorianDate[0];

      var gregorianMonth = gregorianDate[1];

      var gregorianDay = gregorianDate[2];

      // Create a Date object with the converted Gregorian date

      var inputDate = new Date(gregorianYear, gregorianMonth - 1, gregorianDay);

      // Set the time components of inputDate to midnight (00:00:00)

      inputDate.setHours(0, 0, 0, 0);

      // Get the current date

      var currentDate = new Date();

      // Set the time components of currentDate to midnight (00:00:00)

      currentDate.setHours(0, 0, 0, 0);

      // Get the expiration date from the 6th column of the users sheet

      var expirationDate = getUserExpirationDate(userId);

      // Set the time components of expirationDate to midnight (00:00:00)

      expirationDate.setHours(0, 0, 0, 0);

      // Compare the input date with the current date and the expiration date

      return inputDate >= currentDate && inputDate <= expirationDate;

    } catch (error) {

      // Handle conversion errors

      console.error('Error converting solar to Gregorian date:', error);

      Logger.log('Error converting solar to Gregorian date:', error);

      sendMessage(userId, 'Error converting solar to Gregorian date:', error);

      return false;

    }

  }

  return false;

}

function isValidHour(Info) {

  var InfoPattern = /^(0?([0-9]|1[0-9]|2[0-3]))$/;

  return InfoPattern.test(Info) && InfoPattern.exec(Info) !== null;

}

//! OTHER FUNCTIONS ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

function getUserExpirationDate(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(USERS\_SHEET\_NAME);

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

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

    if (data[i][1] === userId && data[i][5] !== "") {

      return new Date(data[i][5]);

    }

  }

  return null; // Return null after checking all rows

}

function combineAndConvertToISO(userId) {

  var sheet = SpreadsheetApp.openById(SHEET\_ID).getSheetByName(GUESTS\_SHEET\_NAME);

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

  var lastRow = data.length - 1; // Get the last row index

  // Find the last record of userId where the 6th column (F) is filled and column 7th column (G) is empty

  for (var i = lastRow; i >= 0; i--) {

    if (data[i][1] === userId && data[i][5] !== "" && data[i][6] === "") {

      var guestDateStr = data[i][4];

      var guestHour = data[i][5];

      try {

        var splitedStr = guestDateStr.split("/");

        var solarYear = Number(splitedStr[2]);

        var solarMonth = Number(splitedStr[1]);

        var solarDay = Number(splitedStr[0]);

        // // Parse guestDateStr into a JavaScript Date object

        // var guestDateObj = new Date(guestDateStr);

        // // Extract components from the Date object

        // var solarYear = guestDateObj.getFullYear();

        // var solarMonth = guestDateObj.getMonth() + 1; // Months are zero-based, so add 1

        // var solarDay = guestDateObj.getDate();

        // Check if date components are valid

        if (isNaN(solarDay) || isNaN(solarMonth) || isNaN(solarYear)) {

          throw new Error('Invalid date components');

        }

        // Parse guestHour as an integer

        var hour = parseInt(guestHour, 10);

        // Check if hour is valid

        if (isNaN(hour) || hour < 0 || hour > 23) {

          throw new Error('Invalid hour');

        }

        // Convert the solar date to Gregorian date

        var gregorianDate = Converter.persianToGregorian(solarYear, solarMonth, solarDay);

        var gregorianYear = gregorianDate[0];

        var gregorianMonth = gregorianDate[1];

        var gregorianDay = gregorianDate[2];

        // Create a JavaScript Date object with the components

        var combinedDate = new Date(gregorianYear, gregorianMonth - 1, gregorianDay, hour, 0, 0, 0);

        // Convert the Date object to ISO format

        var guestISODate = combinedDate.toISOString();

        return guestISODate;

      } catch (error) {

        Logger.log('Error in combineAndConvertToISO: ' + error);

        throw new Error('Error in combineAndConvertToISO: ' + error);

      }

    }

  }

  // If userId is not found or conditions are not met

  return null;

}

//! DATE CONVERTER S2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2GS2G

// Constants

var GREGORIAN\_EPOCH = 1721425.5;

var PERSIAN\_EPOCH = 1948320.5;

// Helper function for modulo operation

function mod(a, b) {

  return ((a % b) + b) % b;

}

// Converter class

function Converter() { }

// Leap Gregorian

Converter.leapGregorian = function (year) {

  return ((year % 4) === 0) && (!((year % 100 === 0) && (year % 400 !== 0)));

};

// Gregorian to Julian

Converter.gregorianToJulian = function (year, month, day) {

  var pad = 0;

  if (month <= 2) {

    pad = 0;

  } else if (Converter.leapGregorian(year)) {

    pad = -1;

  } else {

    pad = -2;

  }

  return (GREGORIAN\_EPOCH - 1) +

    (365 \* (year - 1)) +

    Math.floor((year - 1) / 4) +

    (-Math.floor((year - 1) / 100)) +

    Math.floor((year - 1) / 400) +

    Math.floor((((367 \* month) - 362) / 12) + (pad + day));

};

// Julian to Gregorian

Converter.julianToGregorian = function (jd) {

  var wjd = Math.floor(jd - 0.5) + 0.5;

  var depoch = wjd - GREGORIAN\_EPOCH;

  var quadricent = Math.floor(depoch / 146097);

  var dqc = mod(depoch, 146097);

  var cent = Math.floor(dqc / 36524);

  var dcent = mod(dqc, 36524);

  var quad = Math.floor(dcent / 1461);

  var dquad = mod(dcent, 1461);

  var yindex = Math.floor(dquad / 365);

  var year = (quadricent \* 400) + (cent \* 100) + (quad \* 4) + yindex;

  if (!((cent === 4) || (yindex === 4))) {

    year += 1;

  }

  var yearday = wjd - Converter.gregorianToJulian(year, 1, 1);

  var leapadj;

  if (wjd < Converter.gregorianToJulian(year, 3, 1)) {

    leapadj = 0;

  } else if (Converter.leapGregorian(year)) {

    leapadj = 1;

  } else {

    leapadj = 2;

  }

  var month = Math.floor((((yearday + leapadj) \* 12) + 373) / 367);

  var day = (wjd - Converter.gregorianToJulian(year, month, 1)) + 1;

  return [year, month, day];

};

// Leap Persian

Converter.leapPersian = function (year) {

  if (year === 1403) return true; // Well, algorithms are not perfect \o/

  return (

    (((((year - ((year > 0) ? 474 : 473)) % 2820) + 474) + 38) \* 682) % 2816

  ) < 682;

};

// Persian to Julian

Converter.persianToJulian = function (year, month, day) {

  var epbase = year - ((year >= 0) ? 474 : 473);

  var epyear = 474 + mod(epbase, 2820);

  return day +

    ((month <= 7)

      ? ((month - 1) \* 31)

      : (((month - 1) \* 30) + 6)

    ) +

    Math.floor(((epyear \* 682) - 110) / 2816) +

    ((epyear - 1) \* 365) +

    (Math.floor(epbase / 2820) \* 1029983) + (PERSIAN\_EPOCH - 1);

};

// Julian to Persian

Converter.julianToPersian = function (jd) {

  var njd = Math.floor(jd) + 0.5;

  var depoch = njd - Converter.persianToJulian(475, 1, 1);

  var cycle = Math.floor(depoch / 1029983);

  var cyear = mod(depoch, 1029983);

  var ycycle;

  if (cyear === 1029982) {

    ycycle = 2820;

  } else {

    var aux1 = Math.floor(cyear / 366);

    var aux2 = mod(cyear, 366);

    ycycle = Math.floor(((2134 \* aux1) + (2816 \* aux2) + 2815) / 1028522)

      + aux1 + 1;

  }

  var year = ycycle + (2820 \* cycle) + 474;

  if (year <= 0) {

    year -= 1;

  }

  var yday = (njd - Converter.persianToJulian(year, 1, 1)) + 1;

  var month = (yday <= 186) ? Math.ceil(yday / 31) : Math.ceil((yday - 6) / 30);

  var day = (njd - Converter.persianToJulian(year, month, 1)) + 1;

  return [year, month, day];

};

// Persian to Gregorian

Converter.persianToGregorian = function (year, month, day) {

  var julian = Converter.persianToJulian(year, month, day);

  return Converter.julianToGregorian(julian);

};

// Gregorian to Persian

Converter.gregorianToPersian = function (year, month, day) {

  var julian = Converter.gregorianToJulian(year, month, day);

  return Converter.julianToPersian(julian);

};

//! INTERACTS WITH BOT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

function doPost(e) {

  var contents = JSON.parse(e.postData.contents);

  userId = contents.message.from.id;

  var messageText = contents.message.text;

  // check user is in which state?

  var state = GetUserState(userId);

  try {

    //?  USER  USERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSERUSER

    if (state === States.Unregistered) {

      if (messageText === "/start") {

        storeUserIdInUsersSheet(userId);

        sendMessage(userId, "سلام")

        sendMessage(userId, "من ربات خدمات غیرحضوری کارخانه نوآوری قم (صدران) هستم.")

        sendMessage(userId, "خوش آمدید 🌹")

        sendMessage(userId, "در صورت تمایل به پیش‌ثبت نام در فضای اشتراکی، لطفا نام خود را وارد فرمائید (فقط نام و با حروف فارسی).");

      } else { sendMessage(userId, "لطفا برای شروع از /start استفاده کنید."); }

    } else if (state === States.Registering\_FirstNameReq) {

      if (isValidUserFirstName(messageText)) {

        storeUserFirstName(messageText);

        sendMessage(userId, "نام شما ثبت شد.");

        sendMessage(userId, "لطفا نام خانوادگی خود را وارد کنید.");

      } else {

        sendMessage(userId, "🟡 مطمئن شوید نام را با حروف فارسی و کامل وارد کرده‌اید.");

      }

    } else if (state === States.Registering\_LastNameReq) {

      if (isValidUserLastName(messageText)) {

        storeUserLastName(messageText);

        sendMessage(userId, "نام خانوادگی شما ثبت شد.");

        sendMessage(userId, "لطفا شماره تلفن همراه خود را وارد کنید (۱۱ رقم و با اعداد لاتین).");

      } else {

        sendMessage(userId, "🟡 مطمئن شوید نام خانوادگی‌تان را با حروف فارسی و کامل وارد کرده‌اید.");

      }

    } else if (state === States.Registering\_MobileReq) {

      if (isValidUserMobile(messageText)) {

        storeUserMobile(messageText);

        sendMessage(userId, "🟢 شماره تلفن همراه شما ثبت و پیش‌ثبت نام انجام شد.");

        sendMessage(userId, "با تشکر، درخواست شما توسط همکاران ما بررسی خواهد شد.");

      } else {

        sendMessage(userId, "🟡 مطمئن شوید شماره تلفن را درست، ۱۱ رقمی و با 09 (در ابتدای شماره) وارد کرده‌اید.");

      }

    } else if (state === States.Registering\_WaitForAccept) {

      sendMessage(userId, "ثبت نام شما در دست بررسی است؛ صبور باشید 🌼");

    } else if (state === States.Registered\_Expired) {

      var userExpirationDate = getUserExpirationDate(userId);

      // Convert Gregorian date to Persian (Solar) date

      var persianDate = Converter.gregorianToPersian(userExpirationDate.getFullYear(), userExpirationDate.getMonth() + 1, userExpirationDate.getDate());

      var formattedPersianDate = persianDate[2] + '/' + persianDate[1] + '/' + persianDate[0];

      sendMessage(userId, "🥀 اشتراک شما در تاریخ " + formattedPersianDate + " منقضی شده است. لطفاً نسبت به تمدید آن اقدام فرمائید.");

      //? GUEST  GUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUESTGUEST

    } else if (state === States.Registered) {

      if (messageText === "/account") {

        var userExpirationDate = getUserExpirationDate(userId);

        // Convert Gregorian date to Persian (Solar) date

        var persianDate = Converter.gregorianToPersian(userExpirationDate.getFullYear(), userExpirationDate.getMonth() + 1, userExpirationDate.getDate());

        var formattedPersianDate = persianDate[2] + '/' + persianDate[1] + '/' + persianDate[0];

        sendMessage(userId, "⌚ اشتراک شما در " + formattedPersianDate + " منقضی خواهد شد.");

      } else if (messageText === "/mehman") {

        storeUserIdInGuestsSheet(userId);

        sendMessage(userId, "لطفا کد ملی مهمان خود را وارد کنید (۱۰ رقم و با اعداد لاتین).");

      } else {

        sendMessage(userId, "مشترک گرامی! لطفا جهت دریافت هر یک از خدمات زیر، کد مربوطه را بزنید:");

        sendMessage(userId, "مشاهده اطلاعات اشتراک /account");

        sendMessage(userId, "درخواست حضور مهمان /mehman");

      }

    } else if (state === States.Registered\_InviteGuest\_NCReq) {

      if (isValidNationalCode(messageText)) {

        storeGuestNationalCode(messageText);

        sendMessage(userId, "کد ملی مهمان شما ثبت شد.");

        sendMessage(userId, "لطفا شماره تلفن مهمان‌تان را وارد کنید (۱۱ رقم و با اعداد لاتین).");

      } else { sendMessage(userId, "🟡 مطمئن شوید کد ملی را درست و ۱۰ رقمی وارد کرده‌اید."); }

    } else if (state === States.Registered\_InviteGuest\_MobileReq) {

      if (isValidGuestMobile(messageText)) {

        storeGuestMobile(messageText);

        sendMessage(userId, "شماره تلفن مهمان شما ثبت شد.");

        sendMessage(userId, "لطفا تاریخ حضور مهمان‌تان را وارد کنید... با حروف لاتین و مانند 30/10/1402 (روز ماه سال)");

      } else { sendMessage(userId, "🟡 مطمئن شوید شماره تلفن را درست، ۱۱ رقمی و با 09 (در ابتدای شماره) وارد کرده‌اید."); }

    } else if (state === States.Registered\_InviteGuest\_DateReq) {

      if (isValidDate(messageText)) {

        storeGuestDate(messageText);

        sendMessage(userId, "تاریخ حضور مهمان شما ثبت شد.");

        sendMessage(userId, "لطفا «ساعت» حضور مهمان را وارد کنید (عدد لاتین بین 0 تا 23).");

      } else {

        var userExpirationDate = getUserExpirationDate(userId);

        // Convert Gregorian date to Persian (Solar) date

        var persianDate = Converter.gregorianToPersian(userExpirationDate.getFullYear(), userExpirationDate.getMonth() + 1, userExpirationDate.getDate());

        var formattedPersianDate = persianDate[2] + '/' + persianDate[1] + '/' + persianDate[0];

        sendMessage(userId, "🟡 مطمئن شوید تاریخ حضور مهمان، برای آینده، در طول مدت اشتراک شما (تا " + formattedPersianDate + ") و با قالب 30/12/1402 (روز ماه سال) باشد.");

      }

    } else if (state === States.Registered\_InviteGuest\_HourReq) {

      if (isValidHour(messageText)) {

        storeGuestHour(messageText);

        var guestISODate = combineAndConvertToISO(userId);

        storeGuestISODate(guestISODate);

        sendMessage(userId, "🟢 ساعت حضور مهمان شما ثبت و در خواست تکمیل شد.");

        sendMessage(userId, "ایشان می‌توانند در تاریخ و ساعت تعیین شده، با ارائه کارت ملی به سیستم هوشمند کنترل تردد، وارد مجموعه شوند.");

      } else { sendMessage(userId, "🟡 مطمئن شوید ساعت حضور مهمان، با عدد لاتین بین 0 تا 23 تعیین شده."); }

    } else if (state === States.Unknown) {

      sendMessage(userId, "💔 خطایی رخ داده است! لطفا بعدا امتحان کنید.");

    }

  } catch (error) {

    // sendMessage(userId, "Error in doPost: " + error);

    Logger.log("Error in doPost: " + error);

  }

}

29/1/1403  
آقای سهرابی برای تکمیل پروژه برای روزهای بیشتر از 12 ارور میداد این کارها رو کرد:  
1- وقتی دید که ارور date format components رو میده، فهمید که داخل فانکشن combineAndConvertToISO ، متغیر combinedDate در واقع ترتیب تاریخ میلادی رو داره به صورت آبجکت تاریخ در میاره و بخاطر همین استرینگ تاریخ رو با فانکشن split (که مخصوص GAS هست) تبدیل ساب استرینگ کرد با قرار دادن / اسلش تا روز و ماه و سال جدا جدا هر کدام یک استرینگ شوند.

2- بعد تایپ این ساب استرینگ ها که قاعدتا استرینگ هست رو با فانکشن Number به int تبدیل کرد (چون باز ارور گرفته بودیم که استرینگ مناسب نیست) و ترتیبش رو طبق تاریخ شمسی چید که شد سال ماه روز با ایندکس 2 و 1 و 0 که ایندکس میلادی هست.  
مشکل روزهای 12 به بالا حل شد و مشکل 12 به پایین اومد که combineAndConvertToISO خروجی نمیداد و قفل میکرد.

3- برای فهمیدن اینکه کد تا کجا کار میکنه در جاهای مختلف که فانکشن و قبل و بعد محل کال شدنش sendMessage گذاشت و شماره خط رو نوشت مثلا sendMessage(userId, "L434"); که مثلا یعنی تا خط 434 اجرا شده که به ربات تلگرام میفرستاد، بعد دیدیم که فقط تا

      var guestDateStr = data[i][4]

اجرا میکنه ، فهمید که وقتی میره ستون مربوط به تاریخ رو خوانش کنه با فرمت «آبجکت تاریخ» میخونه پس تایپ ستون تاریخ در هر دو شیت را از automatic به plain text تغییر داد تا برای خوانش تاریخ ارور ندهد و یه استرینگ ساده بشه تا combineAndConvertToISO خروجی بده و کد درست کار کنه. و به این صورت 12 به پایین هم حل شد.