Google Sheet API Usage in C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Google.Apis.Auth.OAuth2;
using Google.Apis.Sheets.v4;
using Google.Apis.Sheets.v4.Data;
using Google.Apis.Services;
using Google.Apis.Util.Store;
using System.IO;
using System.Threading;

namespace DgEeUI.Utils
{
     public class GoogleSheetUtil
     {
          private string _webRootPath;
          private string _spreadSheetId;

          public GoogleSheetUtil(string webRootPath, string spreadSheetId)
          {
               _webRootPath = webRootPath;
               _spreadSheetId = spreadSheetId;
          }
          string[] Scopes = { SheetsService.Scope.Spreadsheets };
          string ApplicationName = "Google Sheets API .NET Quickstart"

          public void RegisterUserInfo(string email,string name)
          {
               UserCredential credential;

               using (var stream =
                   new FileStream(_webRootPath + "\\Content\\GoogleSheetAuth\\" +
"credentials.json", FileMode.Open, FileAccess.Read))
               {
                   // The file token.json stores the user's access and refresh tokens, and is
created
                   // automatically when the authorization flow completes for the first time.
                   string credPath = _webRootPath + "\Content\GoogleSheetAuth\"+"token.json"
                   credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                       GoogleClientSecrets.Load(stream).Secrets,
                       Scopes,
                       "user",
                       CancellationToken.None,
                       new FileDataStore(credPath, true)).Result;
                   Console.WriteLine("Credential file saved to: " + credPath);
               }

               // Create Google Sheets API service.
               var service = new SheetsService(new BaseClientService.Initializer()
               {
                   HttpClientInitializer = credential,
                   ApplicationName = ApplicationName,
               });

               // Define request parameters.
               String spreadsheetId = _spreadSheetId;
               List<ValueRange> updateData = new List<ValueRange>();
               var dataValueRange = new ValueRange();
               dataValueRange.Range= GetRange(service,spreadsheetId);
               var rows = new List<IList<object>>();
```

```
            var values = new List<object>();
            values.Add(name);
            values.Add(email);
            rows.Add(values);
            dataValueRange.Values= rows;
            updateData.Add(dataValueRange);

            BatchUpdateValuesRequest requestBody = new BatchUpdateValuesRequest();
            string valueInputOption = "USER_ENTERED"
            requestBody.ValueInputOption= valueInputOption;
            requestBody.Data= updateData;

            var request = service.Spreadsheets.Values.BatchUpdate(requestBody, spreadsheetId);

            BatchUpdateValuesResponse response = request.Execute();
            Console.WriteLine(response);
        }
        protected static string GetRange(SheetsService service,string sheetId)
        {
            // Define request parameters.
            String spreadsheetId = sheetId;
            String range = "A:A"

            SpreadsheetsResource.ValuesResource.GetRequest getRequest =
                       service.Spreadsheets.Values.Get(spreadsheetId, range);

            ValueRange getResponse = getRequest.Execute();
            IList<IList<Object>> getValues = getResponse.Values;

            int currentCount = getValues.Count() + 1;

            String newRange = "A" + currentCount + ":B"

            return newRange;
        }
    }
}
```