Course: EGDF20
Module: EGE202 Application Programming

Practical 9: Integrating Local & Cloud Storage for Enterprise Application

Objectives: At the end of this lab, the student should be able to implement applications that stores data in local and cloud storage. They will also experiment with container controls such as DataGridView.

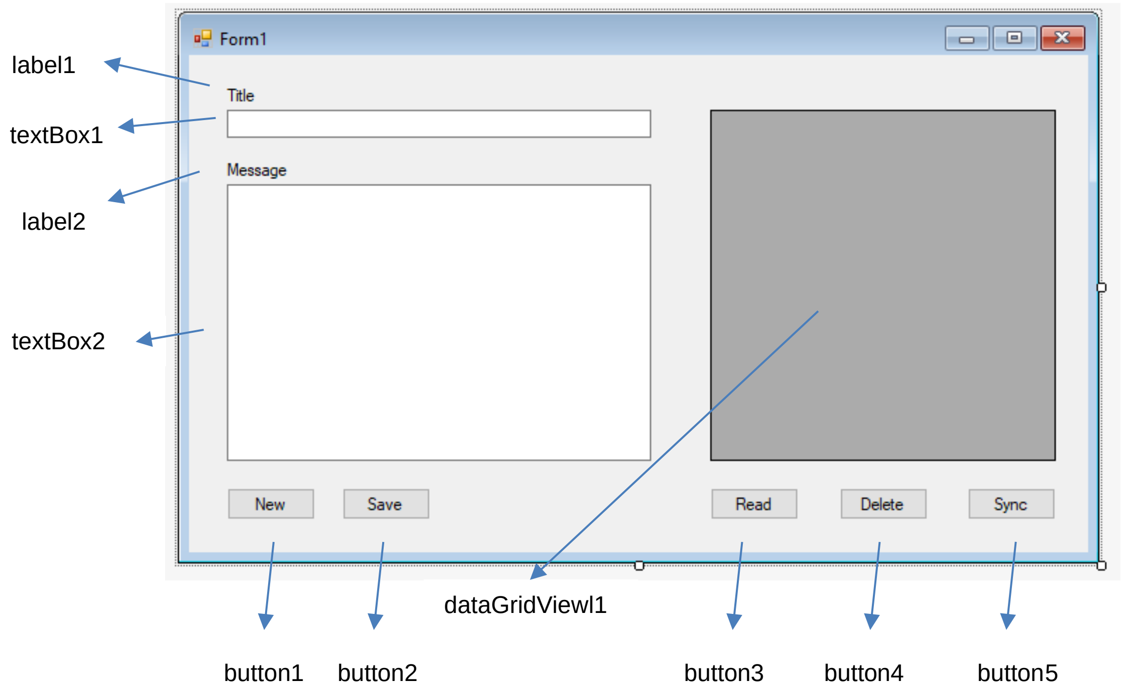
Exercise 1 – Creating a Note Taking Application

Part 1: User Interface Development

1. Under the ***File*** menu, click ***New Project*** or use the ***New Project*** button to create a new project. Alternatively, use the ***Create New Project*** link in the ***Get Started*** *popup* dialog.
2. From the pop-up dialog, select **"C#"** for the ***Language filter***, **"Windows"** for ***the Platform filter*** and **"Desktop"** for the ***Project type filter***.
3. Then choose ***Windows Forms App (.Net Framework)*** and click the ***Next*** button.
4. Type the name of your new project as ***NoteApp*** and keep the Solution name the same as Project name.
5. Set the Location to put the project in your own created folder and finally click on the ***OK*** button.
6. **Do not** tick on the check-box of [□ **Place solution and project in the same directory**].
7. Click the **Create** button to start your project.

8. In the ***Properties*** window of the ***Form*** control, change the ***TopMost*** property of the ***Form1*** to 'True'.
9. Double click on "Form1.cs" ***Solution Explorer*** window to launch the ***Form Designer*** tab.
10. From the ***Toolbar***, drag in 2 *Textbox*, 2 *Label*, 5 *Button* and 1 *DataGridView* control into the *Form1* window. Modify the *(Name)* and *Text* properties based on the table below.

{Name} From	**{Name} To**	**{Text}**	**{MultiLine}**
label1	lblTitle	Title	
label2	lblMesg	Message	
textBox1	txtTitle		
textBox2	txtMesg		True
button1	btnNew	New	
button2	btnSave	Save	
button3	btnRead	Read	
button4	btnDelete	Delete	
button5	btnSync	Sync	
dataGridVew1	dgvMesg		



11. Build and run your application by hitting <F5> or <Ctrl + F5> key. You should see an application with the UI shown in the diagram above.
12. Stop the application.

Part 2: Storing data using a Database Table

1. At the ***Form Designer***, right click and select ***View Code*** to open “Form1.cs”.
2. Add the following code at the starting of the **Form1** class

```
public partial class Form1 : Form
{
    DataTable tbl;

    public Form1()
    {
        InitializeComponent();
    }
```

The DataTable class in C# is a database table representation and provides a collection of columns and rows to store data.

3. In the ***Form Designer*** double click on the ***Form1*** to create *Form1_Load(…)* event handler.

```
private void Form1_Load(object sender, EventArgs e)
{
    tbl = new DataTable("Notes");
    tbl.Columns.Add("Title", typeof(String));
    tbl.Columns.Add("Message", typeof(String));

    dgvMesg.DataSource = tbl;
}
```

4. Now double click on the “Save” ***Button*** in the ***Form Designer***. That will automatically create *btnSave_Click (…)* function.
5. Modify *btnSave_Click (…)* to include the following codes:

```
private void btnSave_Click(object sender, EventArgs e)
{
    if ((txtTitle.Text != String.Empty) && (txtMesg.Text != String.Empty))
        tbl.Rows.Add(txtTitle.Text, txtMesg.Text);
    else
        MessageBox.Show("Title and Message cannot be empty");
}
```

6. Next double click on the “New” ***Button*** in the ***Form Designer***. That will automatically create *btnNew_Click (…)* function.
7. Modify *btnNew_Click (…)* to include the following codes:

```
private void btnNew_Click(object sender, EventArgs e)
{
    txtMesg.Clear();
    txtTitle.Clear();
}
```

8. Build and run your application. Try out the Save and New buttons.
9. Next let’s add the delete function to remove unwanted post.
10. Now double click on the “Delete” ***Button*** in the ***Form Designer***. That will automatically create *btnDelete_Click (…)* function.
11. Modify *btnDelete_Click (…)* to include the following codes:

```
private void btnDelete_Click(object sender, EventArgs e)
{
    int index = dgvMesg.CurrentCell.RowIndex;
    if (index > -1)
    {
         tbl.Rows[index].Delete();
    }
}
```

12. Lastly double click on the “Read” ***Button*** in the ***Form Designer***. That will automatically create *btnRead_Click (…)* function.
13. Modify *btnRead_Click (…)* to include the following codes:

```
private void btnRead_Click(object sender, EventArgs e)
{
    int index = dgvMesg.CurrentCell.RowIndex;

    if (index > -1)
    {
         txtTitle.Text = tbl.Rows[index].ItemArray[0].ToString();
         txtMesg.Text = tbl.Rows[index].ItemArray[1].ToString();
    }
}
```

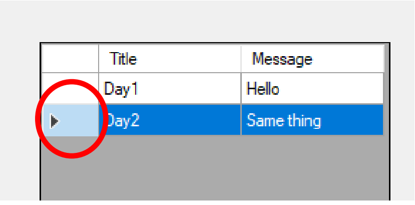
14. Using the ***Build*** menu, select ***Build->Build Solution***. Ensure that there is no error. Run the application to test the Read and Delete function.

No **Actions/Observations**

1

Run the application and try to perform the following

- Resize the column width
- Resize the row height
- Delete the rows by selecting the row header and then using delete key on keyboard



	Title	Message
	Day1	Hello
▸	Day2	Same thing

2

Modify the DataGridView by configuring the following properties in Form Designer.

Control	Property	Value
DataGridVew (dgvMesg)	AllowUserToAddRows	false
	AllowUserToDeleteRows	false
	AllowUserToResizeColumn	false
	AllowUserToResizeRow	false
	ColumnHeadersVisible	false
	RowHeadersVisible	false

Run the application and try to perform the following

- Resize the column width
- Resize the row height
- Delete the rows by selecting the row header and then using delete key on keyboard

3

Coding Task:

Modify the *Form1_Load* as follows:

```
private void Form1_Load(object sender, EventArgs e)
{
   tbl = new DataTable("Notes");
   tbl.Columns.Add("Title", typeof(String));
   tbl.Columns.Add("Message", typeof(String));

   dgvMesg.DataSource = tbl;
   dgvMesg.Columns["Message"].Visible = false;
   dgvMesg.Columns["Title"].Width = dgvMesg.ClientRectangle.Width - 4;
}
```

4	<u>**Coding Task:**</u>
	For database, it is common to implement CRUD functions. C – Create, R- Retrieve, U- Update and D – Delete
	
	Implement Update button and the update functionality
	
	Hint
	`tbl.Rows[index].SetField(0, txtTitle.Text);`
	`tbl.Rows[index].SetField(1, txtMesg.Text);`

Part 3: Persist Data Using Text File

1. So far, each time the application runs, all the previous data is lost because those previous data are not persisted in the disk storage.
2. Using the same solution/project from Part2, double click on the "Sync" ***Button*** in the ***Form Designer***. That will automatically create *btnSync_Click (…)* function.
3. Modify *btnSync_Click (…)* to include the following codes:

```
private void btnSync_Click(object sender, EventArgs e)
{
      tbl.WriteXml(@"MyData.xml");
 }
```

This will save your data in the form of XML data format into your project *bin\debug* folder

4. Build the application, add some entry and click the Sync button. Make sure the file *MyData.xml* is created.
5. Next we will modify *Form1_Load(…)* event handler load the XML data into the ***DataTable*** object *tbl* each time the application starts,

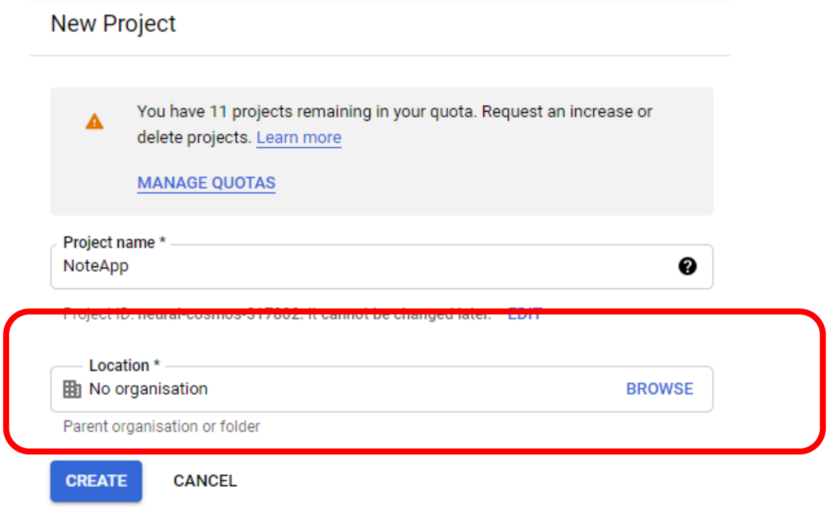
```
private void Form1_Load(object sender, EventArgs e)
{
    tbl = new DataTable("Notes");
    tbl.Columns.Add("Title", typeof(String));
    tbl.Columns.Add("Message", typeof(String));

    if (File.Exists(@"MyData.xml"))
    {
        tbl.ReadXml(@"MyData.xml");
    }

    dgvMesg.DataSource = tbl;
    dgvMesg.Columns["Message"].Visible = false;
    dgvMesg.Columns["Title"].Width = dgvMesg.ClientRectangle.Width - 4;
}
```

6. Build the application and make sure data can now be persisted.

Exercise 2 – Integrating with Cloud Storage (Optional)

Part1: Google Cloud Platform (GCP) Project Preparation

1. Follow instructions at https://developers.google.com/workspace/guides/create-project
2. Open the Google Cloud Console https://console.cloud.google.com/
3. Create a new Project (Leave the organization as No Organization)



New Project

You have 11 projects remaining in your quota. Request an increase or delete projects. Learn more

Project name *: NoteApp

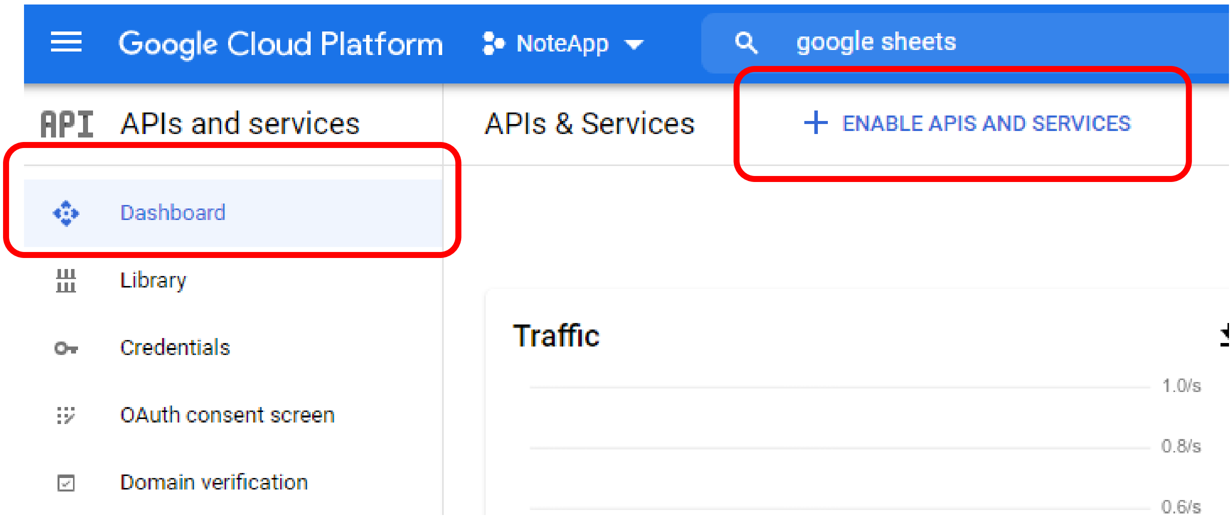
Project ID: neural-cosmos-317802. It cannot be changed later.

Location *: No organisation

Parent organisation or folder

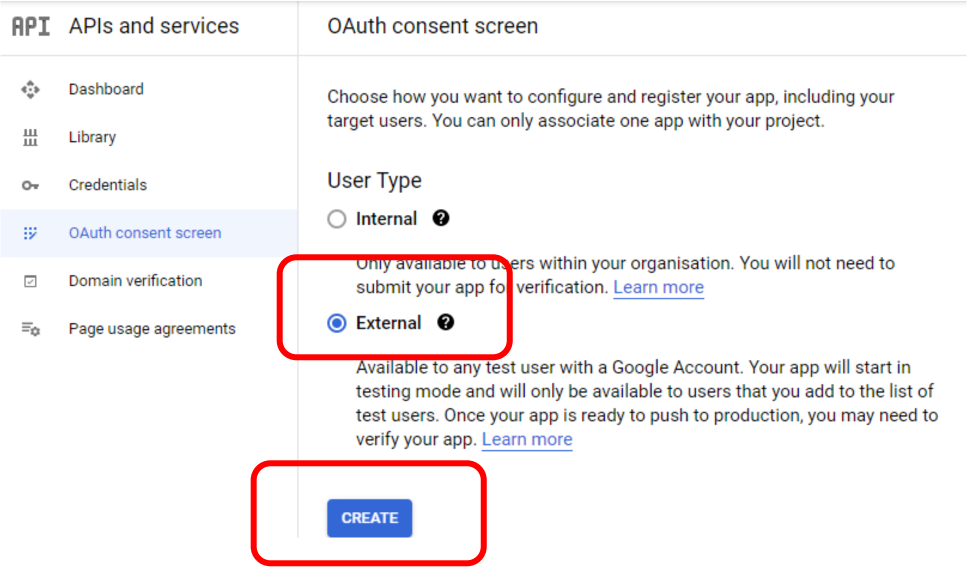
Part 2: Setting up Google Sheets API and Google Authorization Credentials

1. Open the Google Cloud Console https://console.cloud.google.com/
2. From the menu, select ***APIs and Services -> Dashboard***



3. Click + ENABLE APIS AND SERVICE and then search for Google Sheets API and enable it,

4. From the menu, select ***APIs and Services -> OAuth Consent Screen***



Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

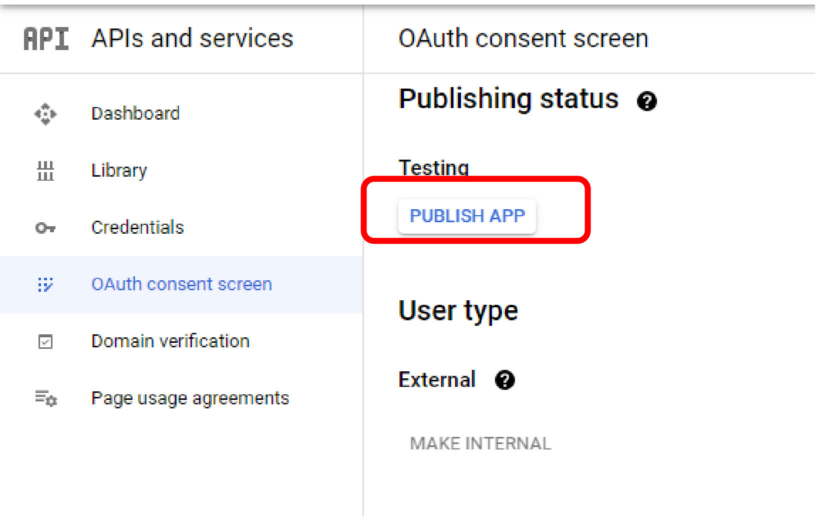
○ Internal

Only available to users within your organisation. You will not need to submit your app for verification. Learn more

● External

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users that you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. Learn more

5. Then select *External* for ***User Type*** and click ***Create***
6. Next enter *NoteApp* for ***App name*** and enter your email address for ***User support email and Developer email address***
7. Then click Next to complete the registration.
8. Next from the menu, select ***APIs and Services -> OAuth Consent Screen*** and click ***Publish App.***



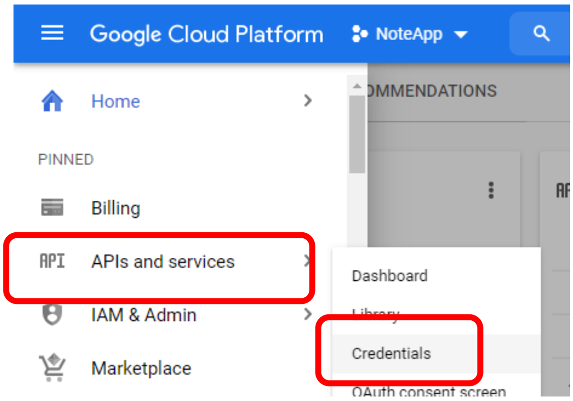
Publishing status

Testing

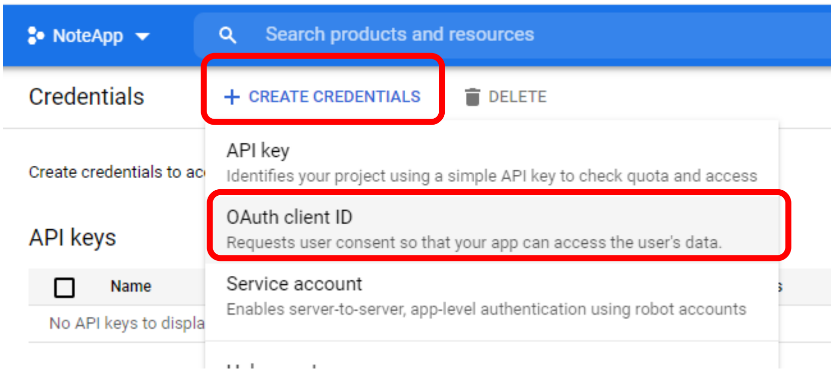
User type

External

9. From the menu, select ***APIs and Services -> Credentials***



10. Then click ***+ CREATE CREDNETIALS*** and select ***OAuth ClientID***

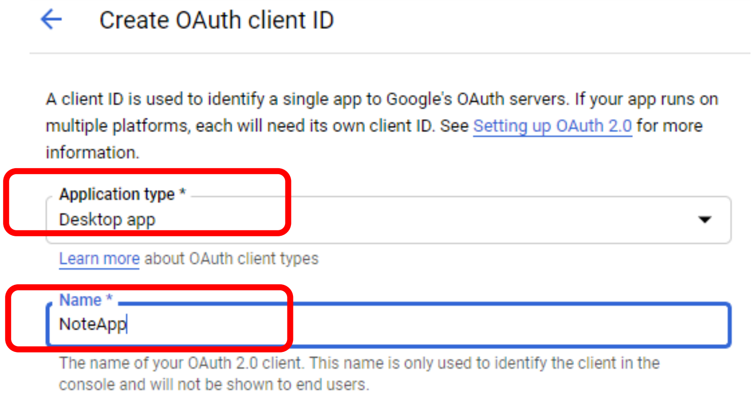


API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so that your app can access the user's data.

Service account
Enables server-to-server, app-level authentication using robot accounts

11. Then set Application Type as ***Desktop App*** and Name as ***NoteApp***


Create OAuth client ID

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information.

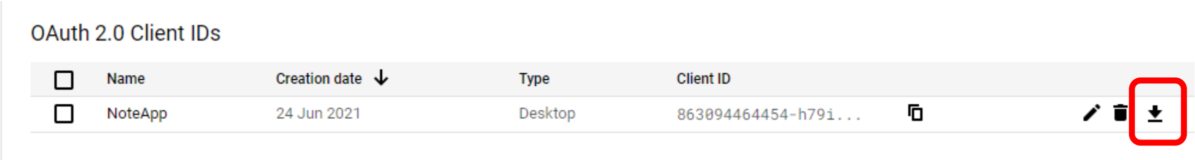
Application type *: Desktop app

Learn more about OAuth client types

Name *: NoteApp

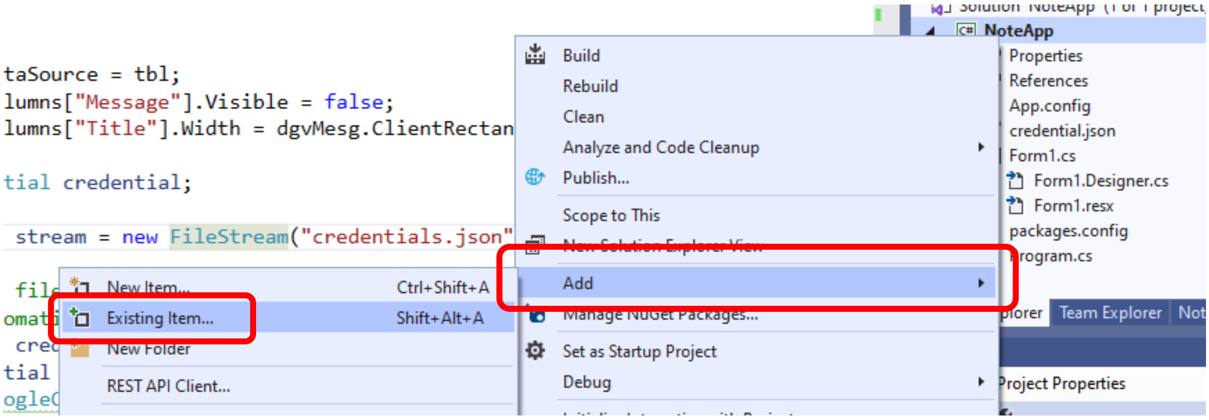
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

12. Finally click the download button and rename the downloaded file as *"credentials.json"*


OAuth 2.0 Client IDs

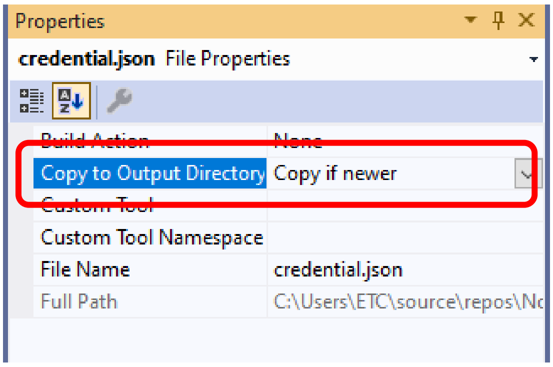
□	Name	Creation date ↓	Type	Client ID
□	NoteApp	24 Jun 2021	Desktop	863094464454-h79i...

13. On ***Solution Explorer***, right click ***NoteApp->Add Existing Item*** and browse to the file *"credentials.json"* and add.



```
taSource = tbl;
lumns["Message"].Visible = false;
lumns["Title"].Width = dgvMesg.ClientRectan

tial credential;

 stream = new FileStream("credentials.json"
```

14. On ***Solution Explorer***, select *"credentials.json"* and right click to access file ***Properties***. Set the ***Copy to Output Directory*** value to ***Copy if newer***.



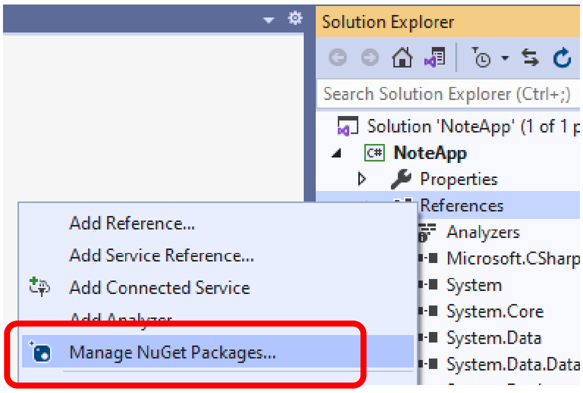
Properties

credential.json File Properties

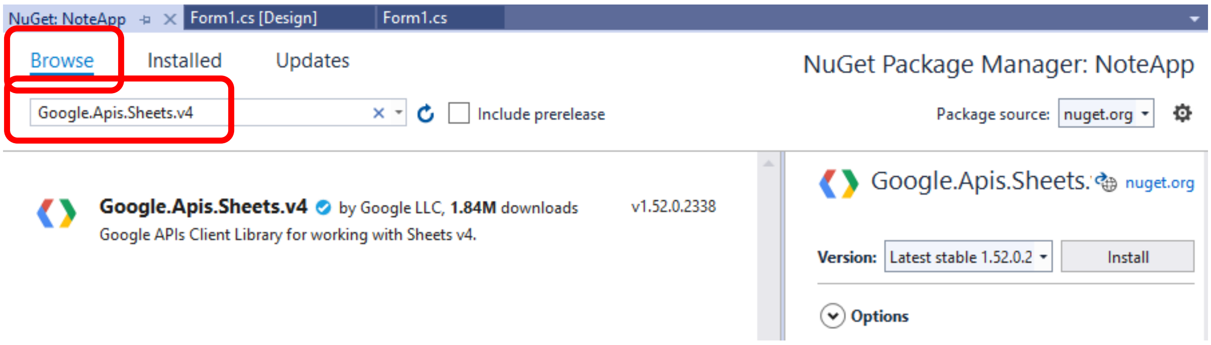
Build Action	None
Copy to Output Directory	Copy if newer
Custom Tool	
Custom Tool Namespace	
File Name	credential.json
Full Path	C:\Users\ETC\source\repos\Nc

Part 3: Setting up Google API Library

1. Using the same solution/project from Part1, at the ***Solution Explorer***, right click References and select ***Manage NuGet Packages***.



2. Then search for “***Google.Apis.Sheets.v4***” in Browse tab and click install.



Google.Apis.Sheets.v4

Package source: nuget.org

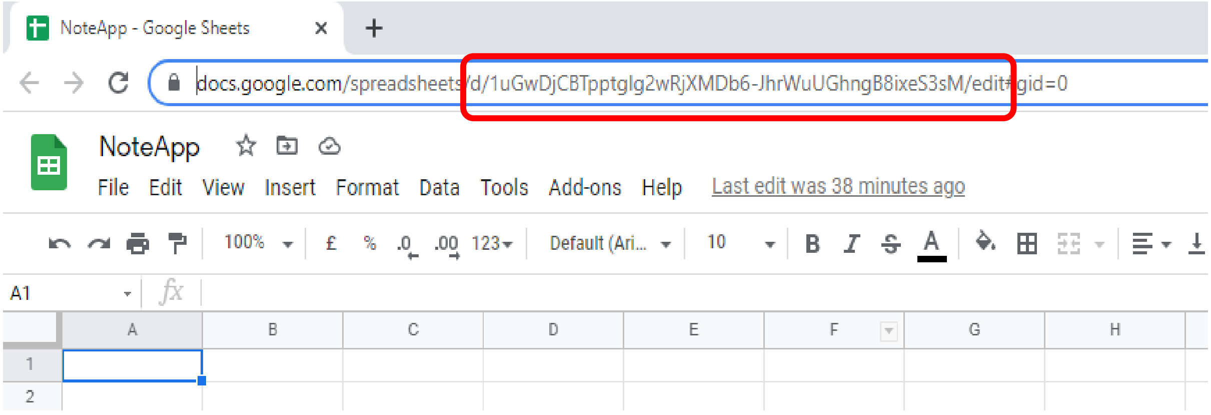
Google.Apis.Sheets.v4 by Google LLC, 1.84M downloads v1.52.0.2338
Google APIs Client Library for working with Sheets v4.

Google.Apis.Sheets. nuget.org

Version: Latest stable 1.52.0.2

Part 4: Accessing Google Sheets

1. In your browser, create a Google Sheet called NoteApp.
2. At the URL, copy the SpreadSheetId to be used in Step 2.



docs.google.com/spreadsheets/d/1uGwDjCBTpptglg2wRjXMDb6-JhrWuUGhngB8ixeS3sM/edit#gid=0

3. Add the following code at the starting of the **Form1** class

```
public partial class Form1 : Form
{
    DataTable tbl;

     static string[] Scopes = { SheetsService.Scope.Spreadsheets };
     static string ApplicationName = "NoteApp";
     String spreadsheetId = "Insert your Spreadsheet ID here";
     SheetsService service;

    public Form1()
    {
        InitializeComponent();
    }
```

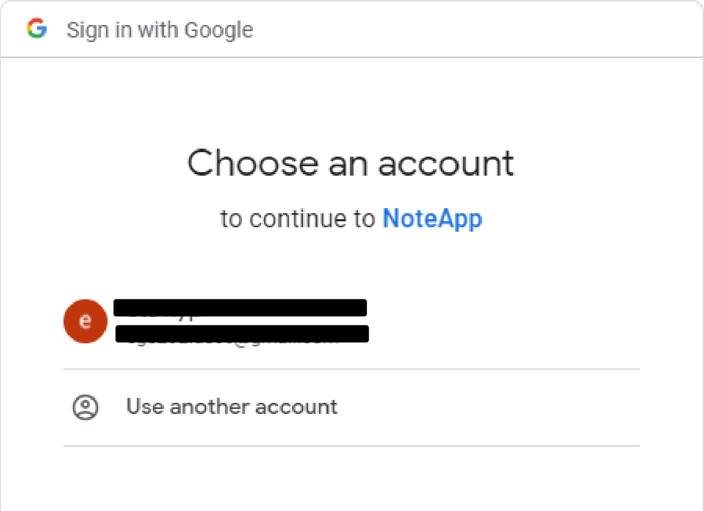
4. Next we will modify *Form1_Load(…)* event handler to

```
    private void Form1_Load(object sender, EventArgs e)
    {
        tbl = new DataTable("Notes");
        tbl.Columns.Add("Title", typeof(String));
        tbl.Columns.Add("Message", typeof(String));

       if (File.Exists(@"MyData.xml"))
       {
            tbl.ReadXml(@"MyData.xml");
       }

       dgvMesg.DataSource = tbl;
       dgvMesg.Columns["Message"].Visible = false;
       dgvMesg.Columns["Title"].Width = dgvMesg.ClientRectangle.Width - 4;


       UserCredential credential;

       using (var stream = new FileStream("credentials.json", FileMode.Open,
FileAccess.Read))
       {
            string credPath = "token.json";
            credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
                 GoogleClientSecrets.FromStream(stream).Secrets,
                 Scopes,
                 "user",
                 CancellationToken.None,
                 new FileDataStore(credPath, true)).Result;
        }

        service = new SheetsService(new BaseClientService.Initializer()
        {
            HttpClientInitializer = credential,
            ApplicationName = ApplicationName,
        });

}
```

5. Modify *btnSync_Click (…)* to include the following codes:

```
   private void btnSync_Click(object sender, EventArgs e)
   {
       tbl.WriteXml(@"MyData.xml");

       ClearValuesRequest cvr = new ClearValuesRequest();
       String range = "Sheet1!A1:B";
       SpreadsheetsResource.ValuesResource.ClearRequest request =
service.Spreadsheets.Values.Clear(cvr, spreadsheetId, range);
       ClearValuesResponse response = request.Execute();

       List<IList<Object>> data = new List<IList<Object>>();
       foreach (DataRow dc in tbl.Rows)
       {
           data.Add(new List<object>(){dc[0].ToString(), dc[1].ToString()});
       }

        ValueRange valueRange = new ValueRange();
        valueRange.MajorDimension = "ROWS";
        range = "Sheet1!A1:B";
        valueRange.Values = data;

        SpreadsheetsResource.ValuesResource.UpdateRequest update =
service.Spreadsheets.Values.Update(valueRange, spreadsheetId, range);
        update.ValueInputOption =
SpreadsheetsResource.ValuesResource.UpdateRequest.ValueInputOptionEnum.RAW;
        UpdateValuesResponse response1 = update.Execute();
    }
```

6. Build and run the application.
7. At first run, you need to authorize the use of the Google Sheets access.

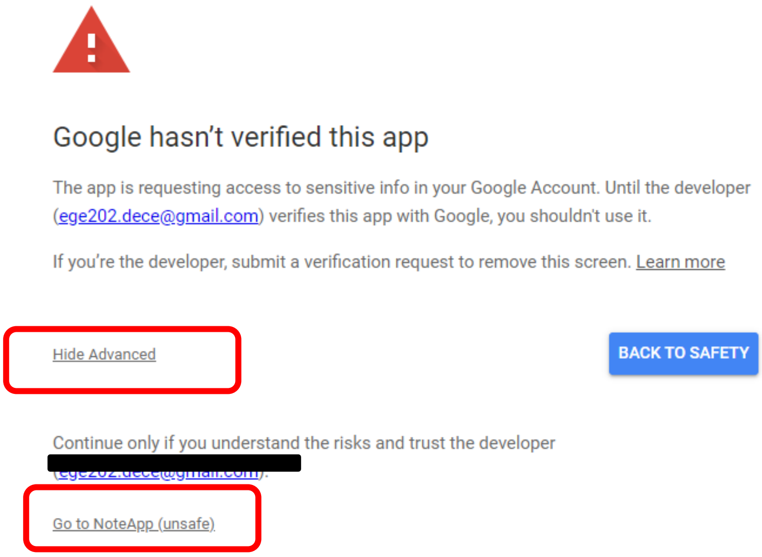


Sign in with Google

Choose an account

to continue to NoteApp

[redacted]

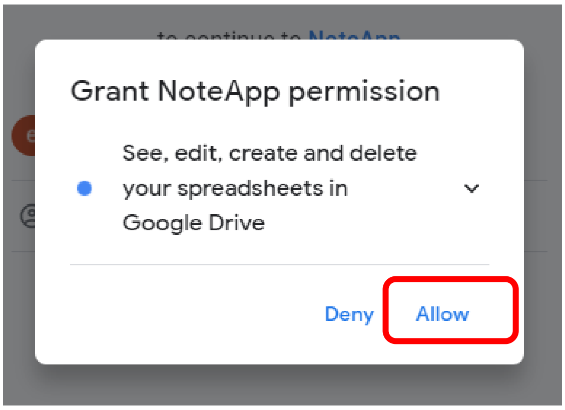


Google hasn't verified this app

The app is requesting access to sensitive info in your Google Account. Until the developer (ege202.dece@gmail.com) verifies this app with Google, you shouldn't use it.

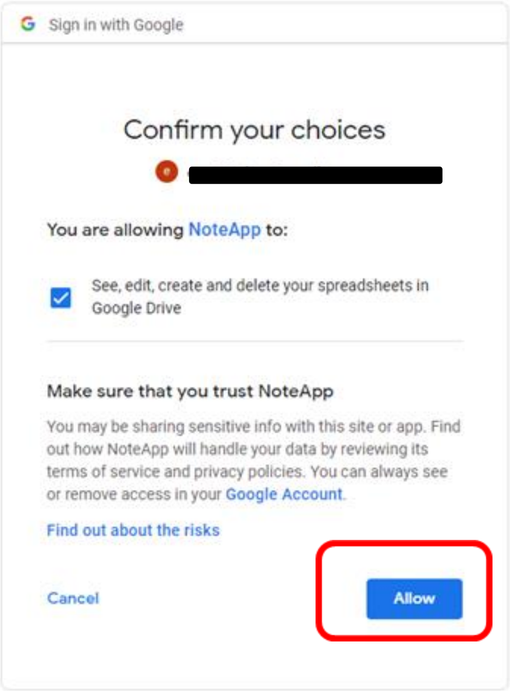
If you're the developer, submit a verification request to remove this screen. Learn more

Continue only if you understand the risks and trust the developer [redacted]



Grant NoteApp permission

● See, edit, create and delete your spreadsheets in Google Drive



Confirm your choices

[redacted]

You are allowing NoteApp to:

☑ See, edit, create and delete your spreadsheets in Google Drive

Make sure that you trust NoteApp

You may be sharing sensitive info with this site or app. Find out how NoteApp will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Find out about the risks