



**TGH**

Making Integrations Simpler



# Integrating With Google Sheets (MuleSoft)

**Author**  
**Asish Palatasingh**

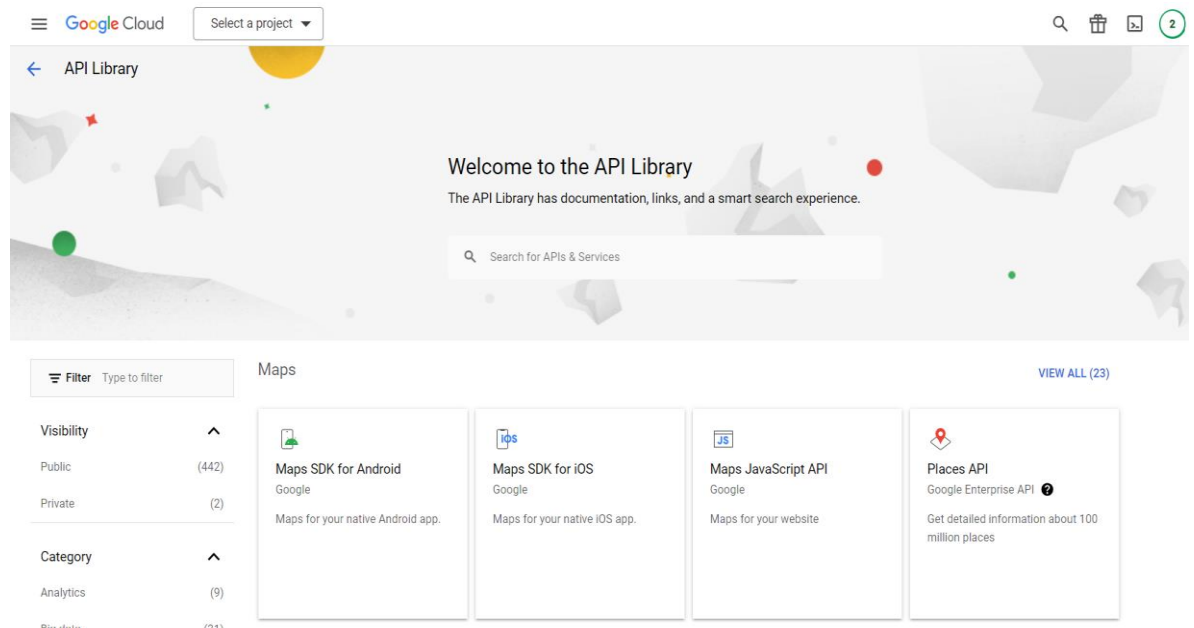


# Google Sheet Module (MuleSoft)

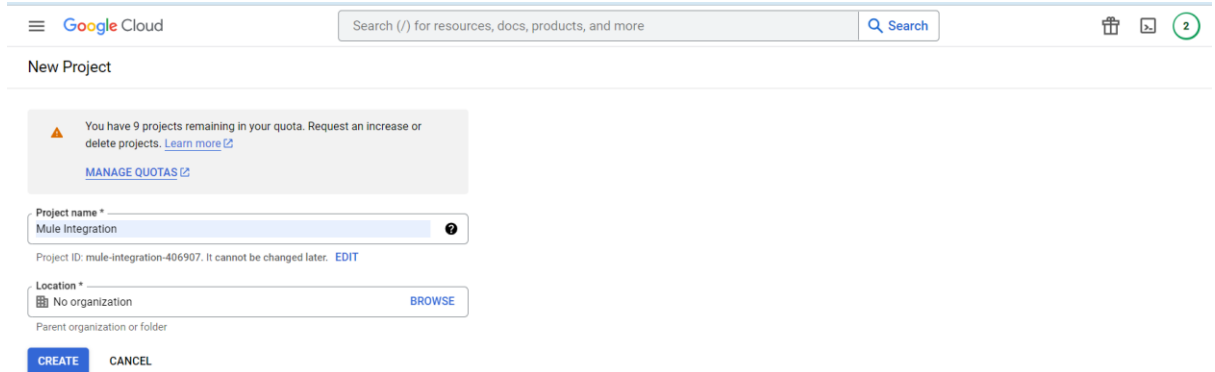
## Prerequisites:

- We need to have an active Google account and we need to make use of Google APIs to leverage the Google Sheet module provided by MuleSoft.
- To do so we need to follow the following steps:
  1. First we need to go to the Google API console using the URL

<https://console.cloud.google.com/apis/library>.

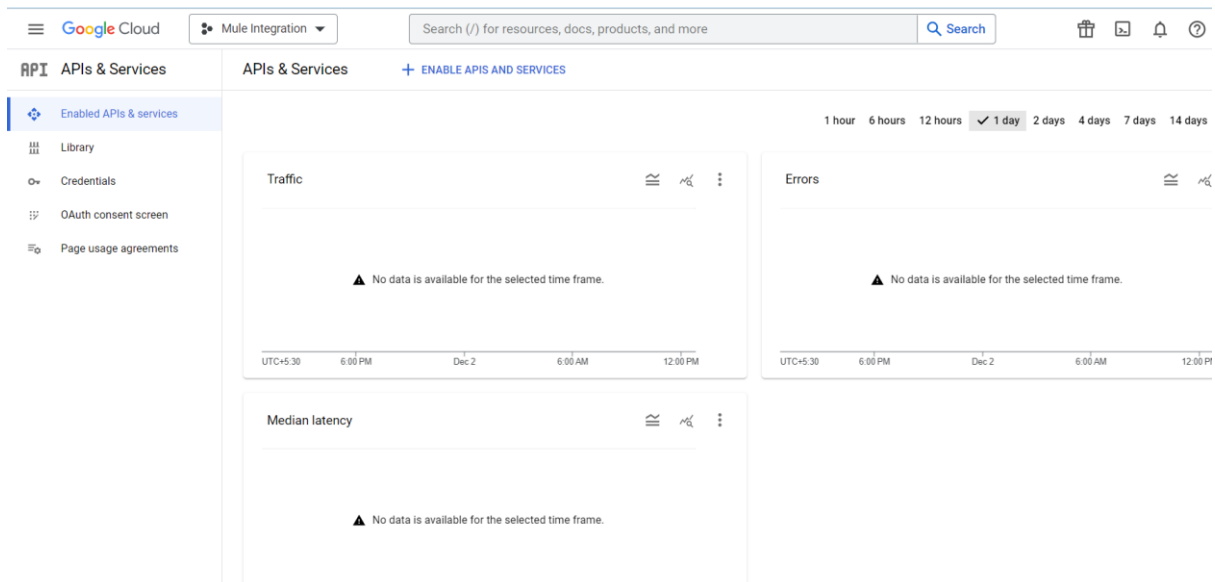


2. Now we need to create a new project by clicking on the select project tab. To create a new project we need to provide a name for the project.



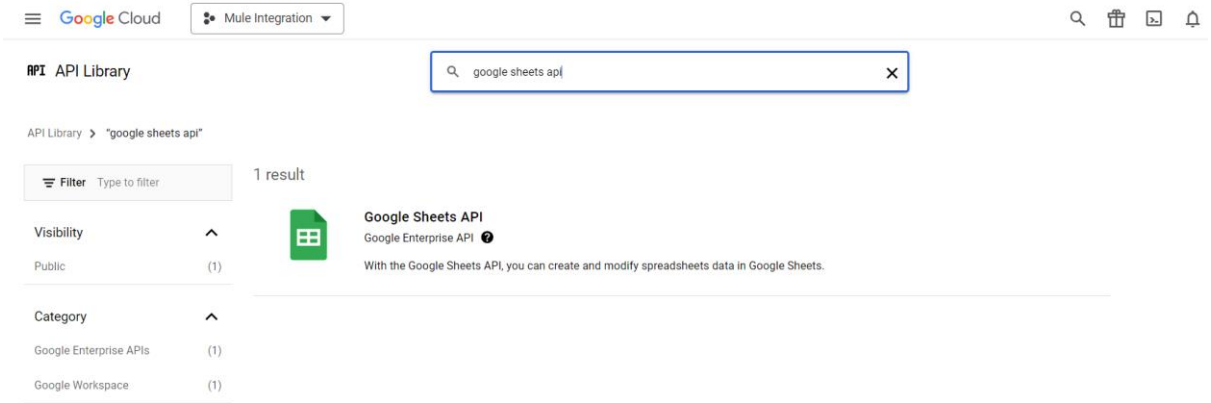
The screenshot shows the 'New Project' page in Google Cloud. At the top, there is a search bar and a notification icon. Below the header, a warning message states: 'You have 9 projects remaining in your quota. Request an increase or delete projects. [Learn more](#) [MANAGE QUOTAS](#)'. The 'Project name' field is filled with 'Mule Integration'. Below it, the 'Project ID' is 'mule-integration-406907'. The 'Location' is set to 'No organization'. At the bottom, there are 'CREATE' and 'CANCEL' buttons.

3. Once the project is created we need to enable the API and Services for the project. To do so we need to click the Enable API and Services.



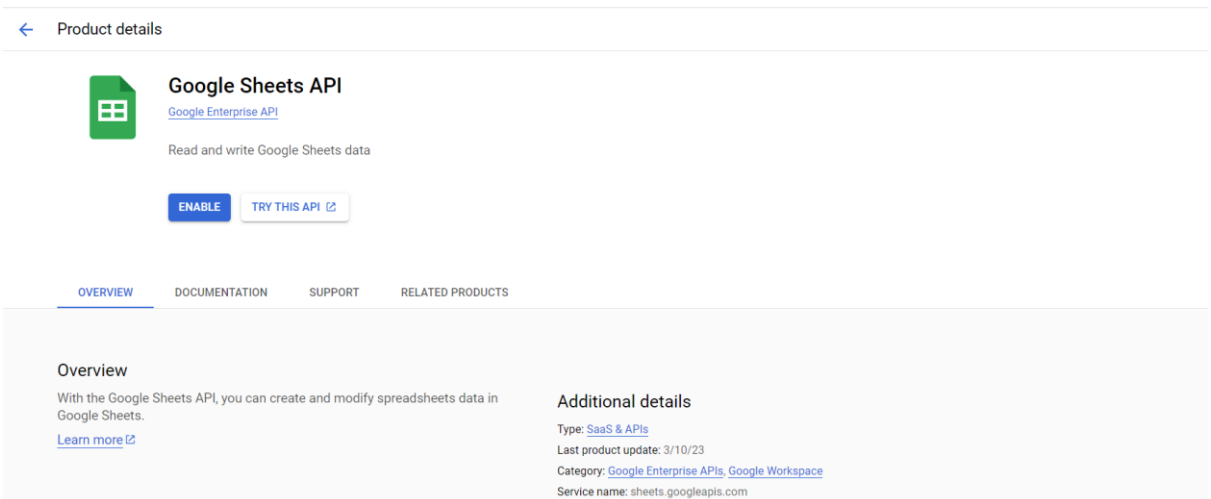
The screenshot shows the 'APIs & Services' page in Google Cloud. The page title is 'APIs & Services' with a '+ ENABLE APIS AND SERVICES' button. On the left, there is a sidebar with 'Enabled APIs & services' selected. The main content area shows three charts: 'Traffic', 'Errors', and 'Median latency'. Each chart has a message: '▲ No data is available for the selected time frame.' The time range is set to '1 day'.

4. We need to search for Google Sheets API to configure it so that we can leverage the Google Sheets API services.



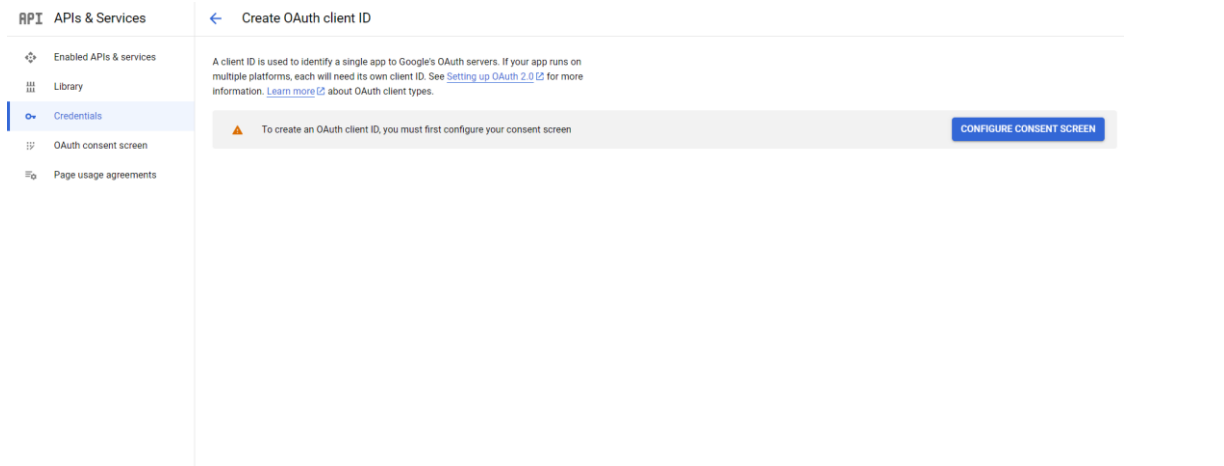
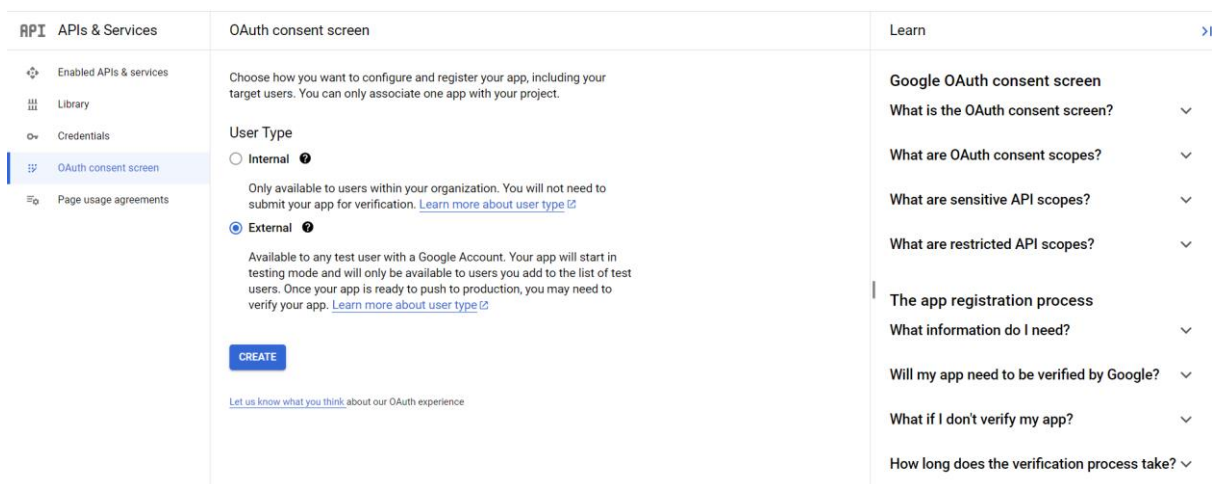
The screenshot shows the Google Cloud API Library interface. At the top, there's a search bar with 'google sheets api' entered. Below the search bar, the results show '1 result'. The result is 'Google Sheets API', which is a 'Google Enterprise API'. The description states: 'With the Google Sheets API, you can create and modify spreadsheets data in Google Sheets.' On the left side, there are filter options for 'Visibility' (Public, 1) and 'Category' (Google Enterprise APIs, 1; Google Workspace, 1).

5. We need to enable the Google Sheets API for our project by clicking on the enable tab.



The screenshot shows the 'Product details' page for the Google Sheets API. The page title is 'Google Sheets API' and it is identified as a 'Google Enterprise API'. The description is 'Read and write Google Sheets data'. There are two buttons: 'ENABLE' and 'TRY THIS API'. Below the buttons are tabs for 'OVERVIEW', 'DOCUMENTATION', 'SUPPORT', and 'RELATED PRODUCTS'. The 'OVERVIEW' tab is selected. The 'Overview' section contains the text: 'With the Google Sheets API, you can create and modify spreadsheets data in Google Sheets. [Learn more](#)'. The 'Additional details' section lists: 'Type: SaaS & APIs', 'Last product update: 3/10/23', 'Category: [Google Enterprise APIs](#), [Google Workspace](#)', and 'Service name: sheets.googleapis.com'.

- To configure the OAuth Client ID we need to configure the OAuth consent screen. Hit on the OAuth consent screen and choose the User Type as External and hit on Create.

- Now we need to create an app for the OAuth consent screen.
  - First we need to fill up the App Name, User support Email, and the email address in the Developer Contact Information and hit Save and Continue.

API APIs & Services

Enabled APIs & services  
Library  
Credentials  
OAuth consent screen  
Page usage agreements

### Edit app registration

1 OAuth consent screen — 2 Scopes — 3 Test users — 4 Summary

#### App information

This shows in the consent screen, and helps end users know who you are and contact you

App name \*  
Mule Integration  
The name of the app asking for consent

User support email \*  
asishpalatasingh2@gmail.com  
For users to contact you with questions about their consent. [Learn more](#)

#### App logo

This is your logo. It helps people recognize your app and is displayed on the OAuth consent screen.  
After you upload a logo, you will need to submit your app for verification unless the app is configured for internal use only or has a publishing status of "Testing". [Learn more](#)

Logo file to upload [BROWSE](#)  
Upload an image, not larger than 1MB on the consent screen that will help users recognize

Application privacy policy link  
Provide users a link to your public privacy policy

Application terms of service link  
Provide users a link to your public terms of service

#### Authorized domains

When a domain is used on the consent screen or in an OAuth client's configuration, it must be pre-registered here. If your app needs to go through verification, please go to the [Google Search Console](#) to check if your domains are authorized. [Learn more](#) about the authorized domain limit.

[+ ADD DOMAIN](#)

#### Developer contact information

Email addresses \*  
asishpalatasingh2@gmail.com  
These email addresses are for Google to notify you about any changes to your project.

[SAVE AND CONTINUE](#) CANCEL

- Then we must provide the necessary scopes for our application and hit Save and Continue.

Credentials  
OAuth consent screen  
Page usage agreements

[ADD OR REMOVE SCOPES](#)

#### Your non-sensitive scopes

API ↑	Scope	User-facing description
No rows to display		

#### Your sensitive scopes

Sensitive scopes are scopes that request access to private user data.

API ↑	Scope	User-facing description
Google Sheets API	../auth/spreadsheets	See, edit, create, and delete all your Google Sheets spreadsheets

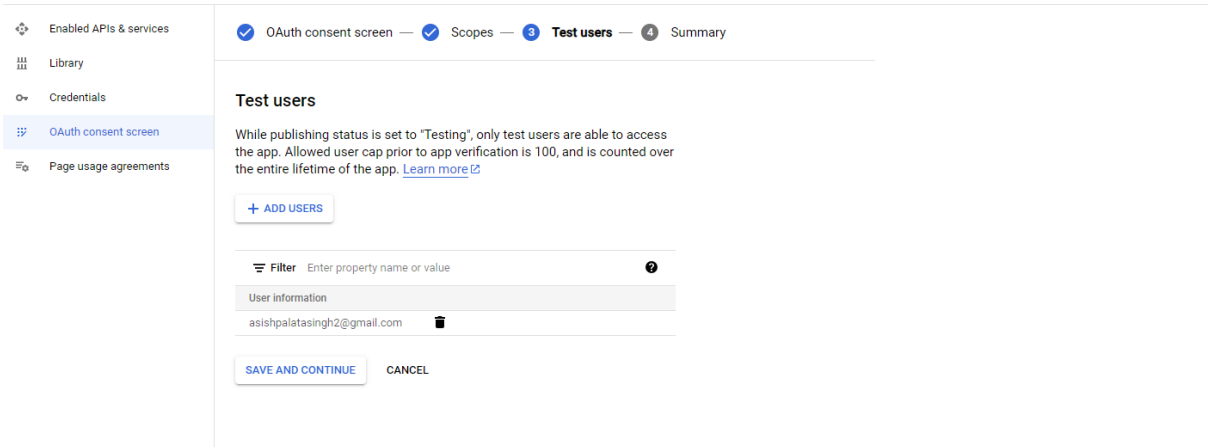
#### Your restricted scopes

Restricted scopes are scopes that request access to highly sensitive user data.

API ↑	Scope	User-facing description
No rows to display		

[SAVE AND CONTINUE](#) CANCEL

- Then we need to add a Test User for our application and hit Save and Continue.



Enabled APIs & services

Library

Credentials

OAuth consent screen

Page usage agreements

OAuth consent screen — Scopes — **Test users** — Summary

### Test users

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

[+ ADD USERS](#)

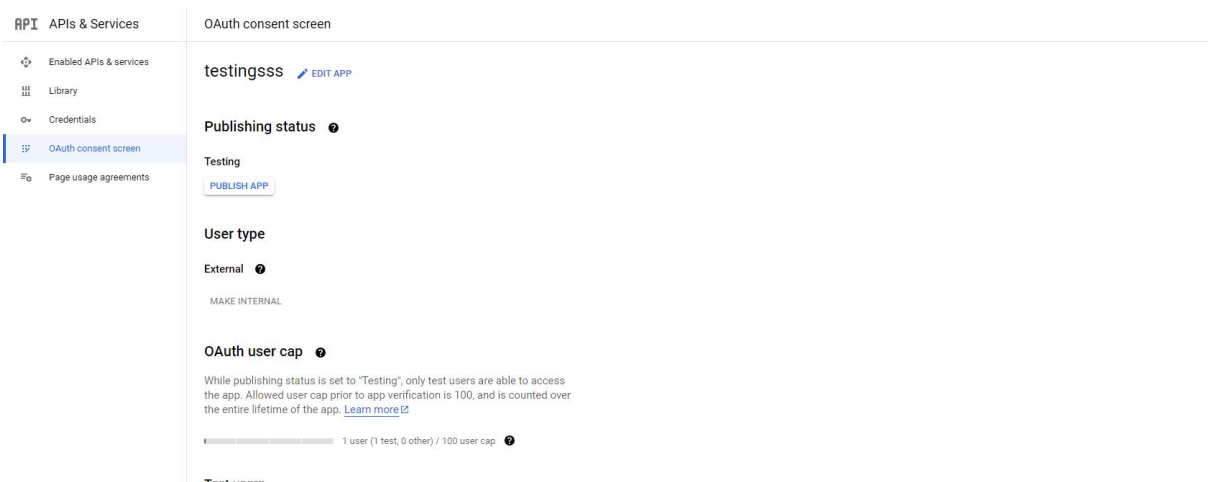
Filter Enter property name or value

User information

asishpalatasingh2@gmail.com

[SAVE AND CONTINUE](#) CANCEL

- Then hit Back To Dashboard and we can see our app.



API APIs & Services

OAuth consent screen

Enabled APIs & services

Library

Credentials

OAuth consent screen

Page usage agreements

testingssss [EDIT APP](#)

**Publishing status**

Testing

[PUBLISH APP](#)

**User type**

External

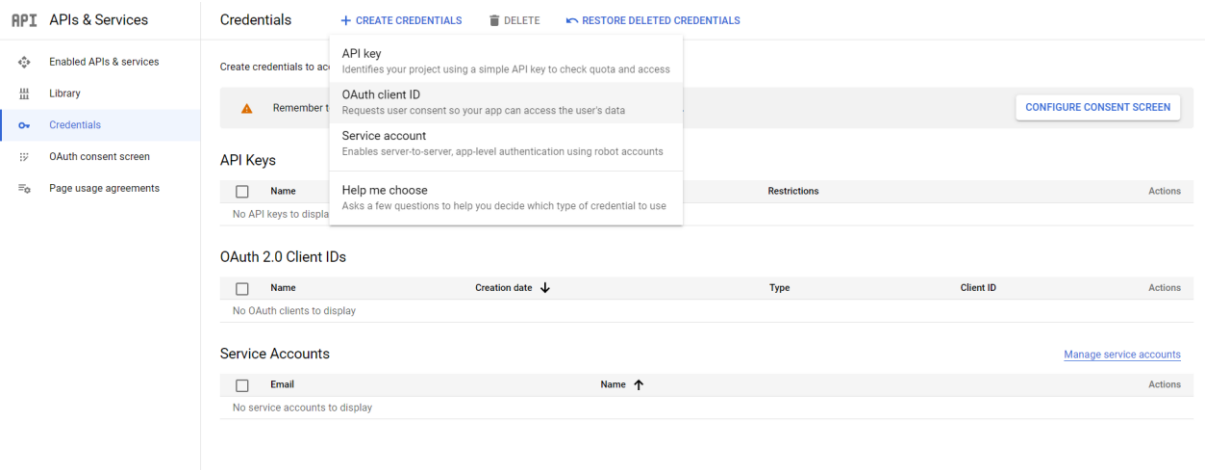
MAKE INTERNAL

**OAuth user cap**

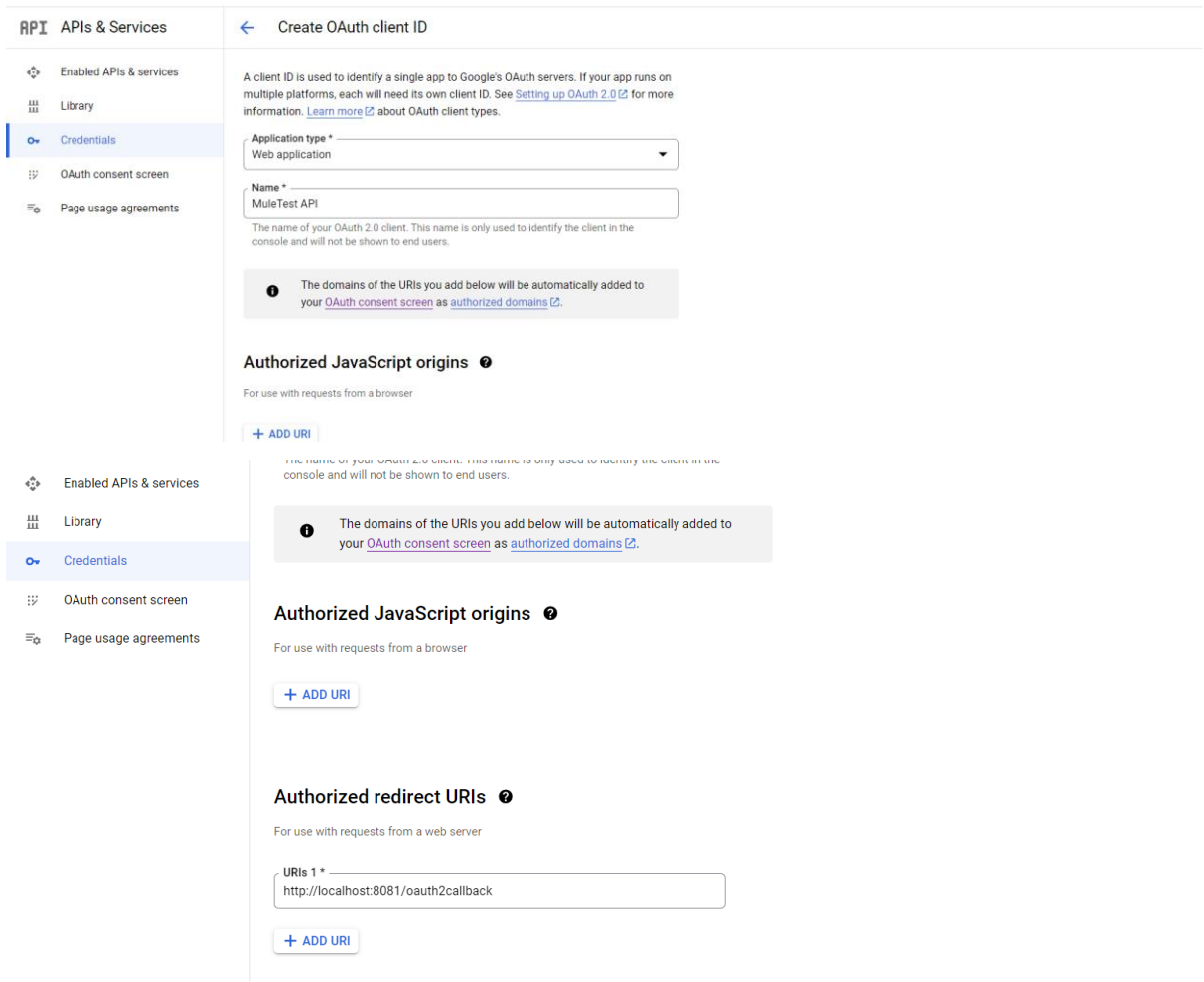
While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

1 user (1 test, 0 other) / 100 user cap

- Now we can see our app in the OAuth consent screen.
- We need to create credentials for our project to get the client id and client secret. To do so we need to hit on the create credentials and choose "OAuth Client ID".

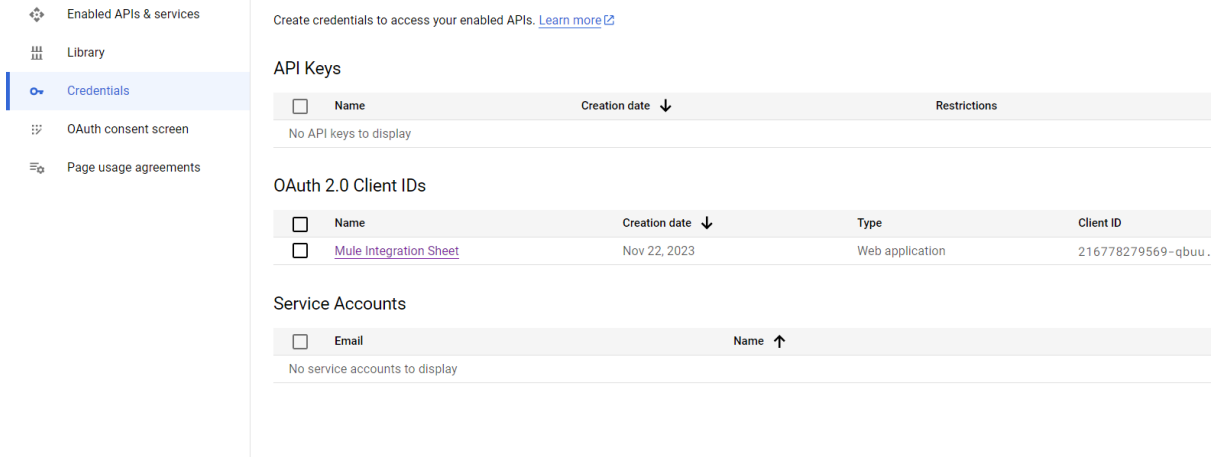


10. We need to provide the Application Type as “Web Application” and we need to provide a name for the application. Then we need to configure the redirect Uri in the Authorized Redirect URI as <http://localhost:8081/oauth2callback>.

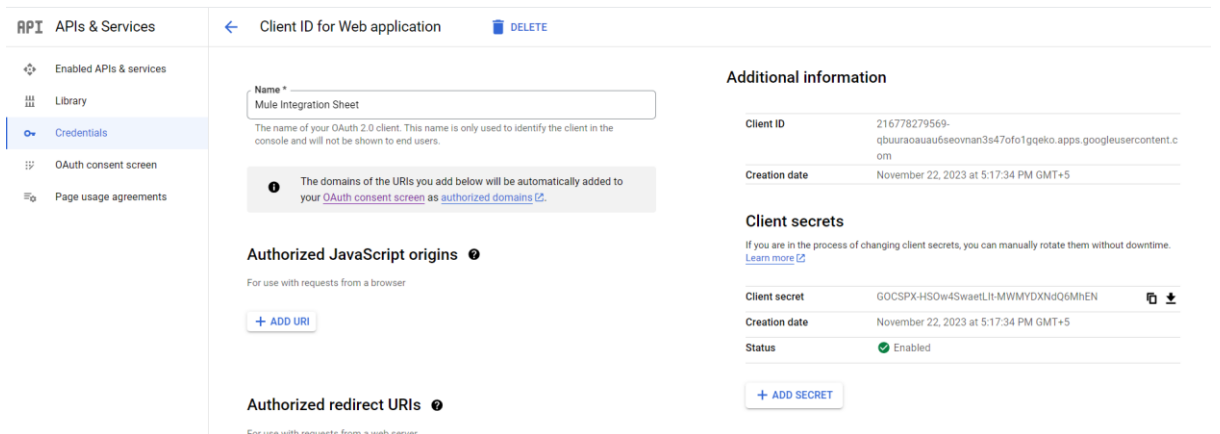




11. We can see the OAuth Client ID under OAuth 2.0 Client IDs and by clicking on it we can get the Client ID and Client Secret which will be used later in the Google Sheets Module.

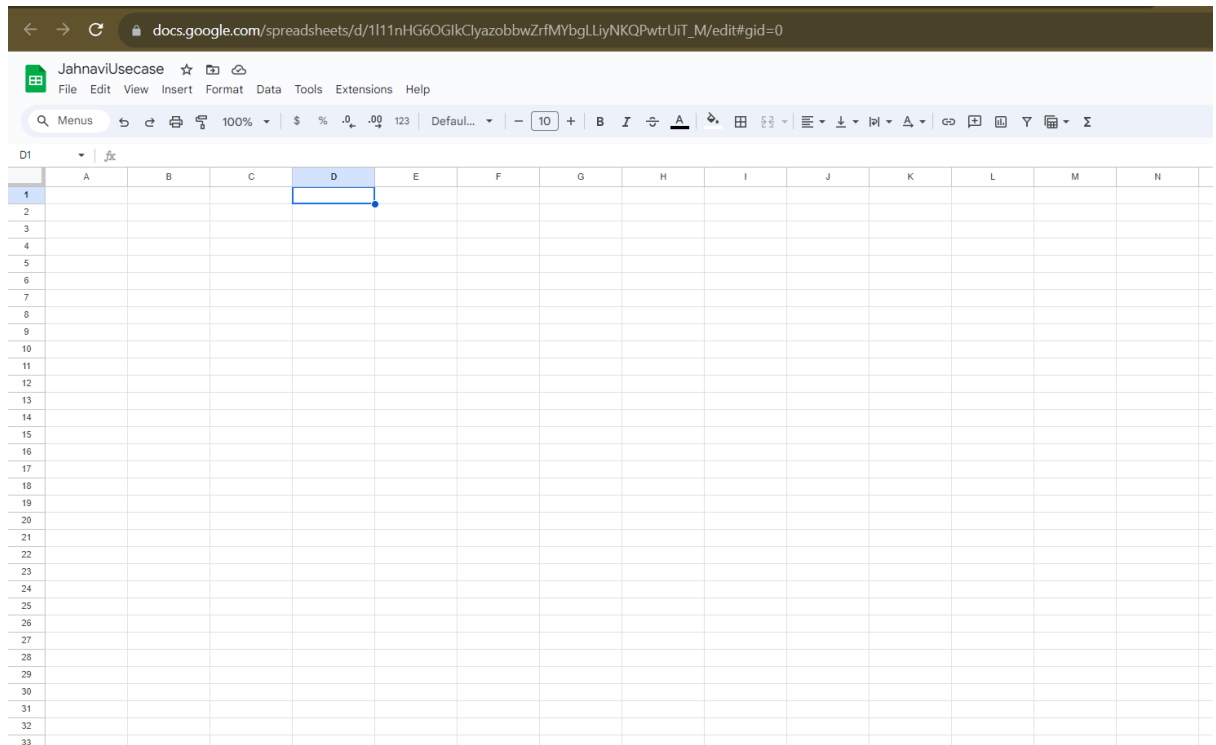


The screenshot shows the 'Credentials' page in the Google Cloud Console. The left sidebar contains navigation options: 'Enabled APIs & services', 'Library', 'Credentials' (selected), 'OAuth consent screen', and 'Page usage agreements'. The main content area is titled 'Create credentials to access your enabled APIs. [Learn more](#)'. It features three sections: 'API Keys' (empty), 'OAuth 2.0 Client IDs' (containing one entry 'Mule Integration Sheet' with creation date 'Nov 22, 2023' and type 'Web application'), and 'Service Accounts' (empty).



The screenshot shows the configuration page for a 'Client ID for Web application'. The left sidebar is the same as the previous screenshot. The main content area has a title 'Client ID for Web application' and a 'DELETE' button. It includes a 'Name' field with the value 'Mule Integration Sheet'. Below this is a warning icon and text: 'The domains of the URIs you add below will be automatically added to your OAuth consent screen as [authorized domains](#).' There are three sections for configuration: 'Authorized JavaScript origins' (with an '+ ADD URI' button), 'Authorized redirect URIs' (with an '+ ADD URI' button), and 'Additional information' (containing 'Client ID', 'Creation date', 'Client secret', and 'Status').

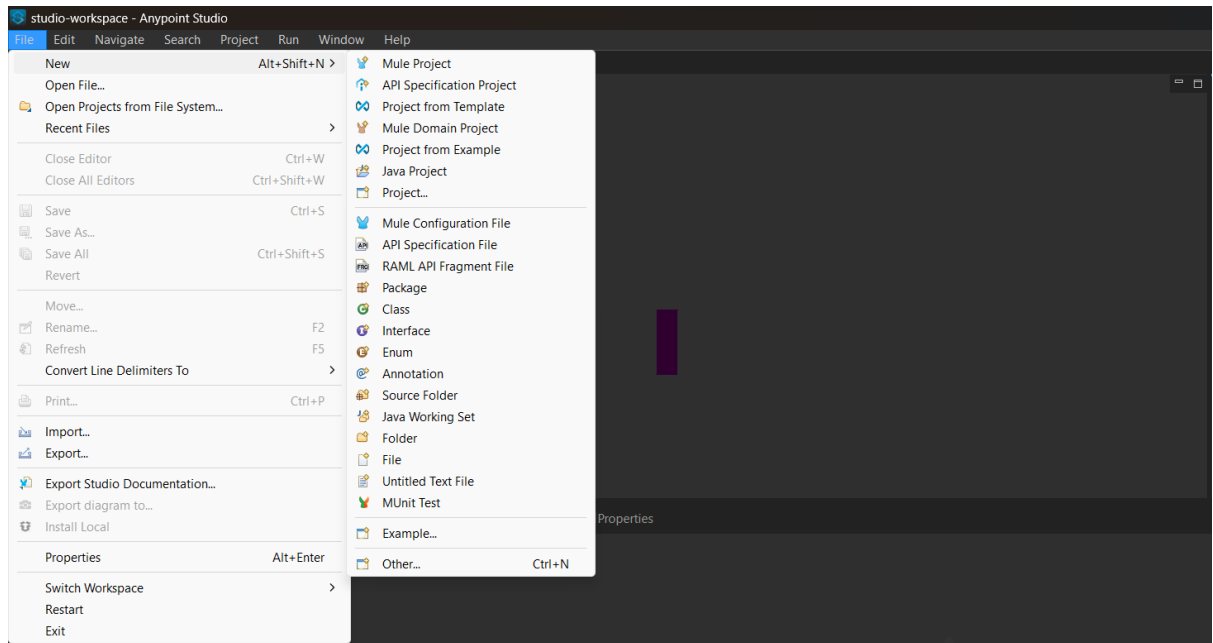
12. We need to create one Google Spread Sheet under the same Email ID that we configured in our Google API. In the URL we can find the Spreadsheet ID which will be used later.



- Now let's configure our Mule application to use the Append Spreadsheet Values from the GoogleSheet Module.

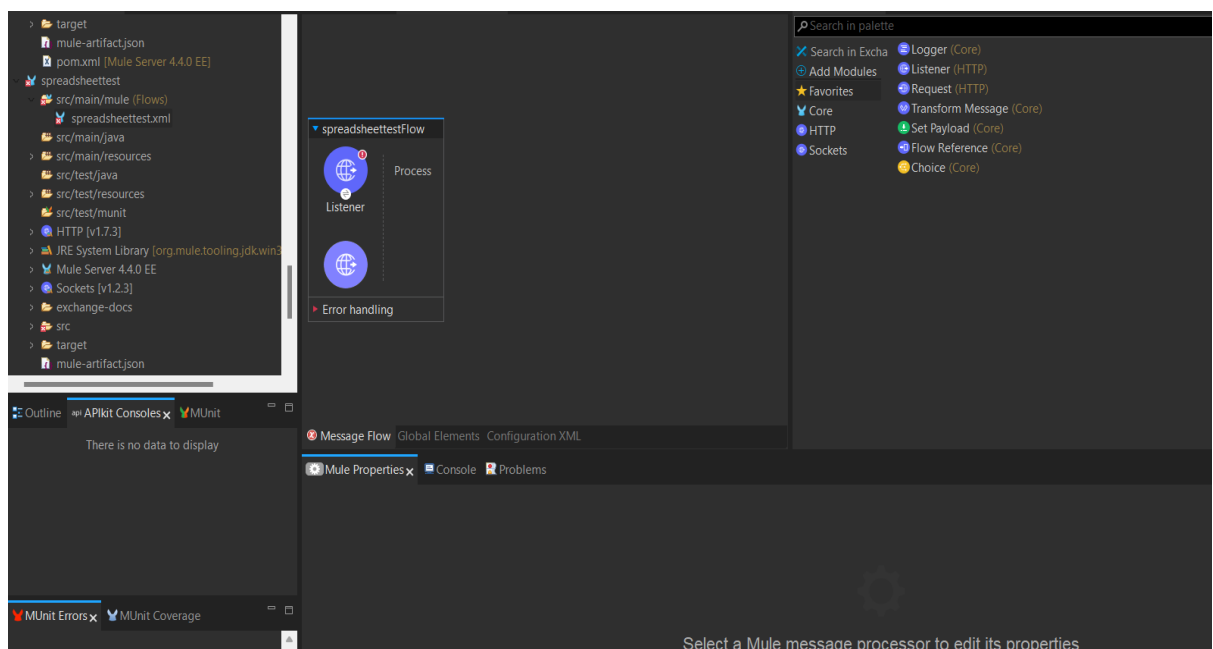
## Step 1:

- Open the Anypoint Studio create a Mule project and provide a name for the project.



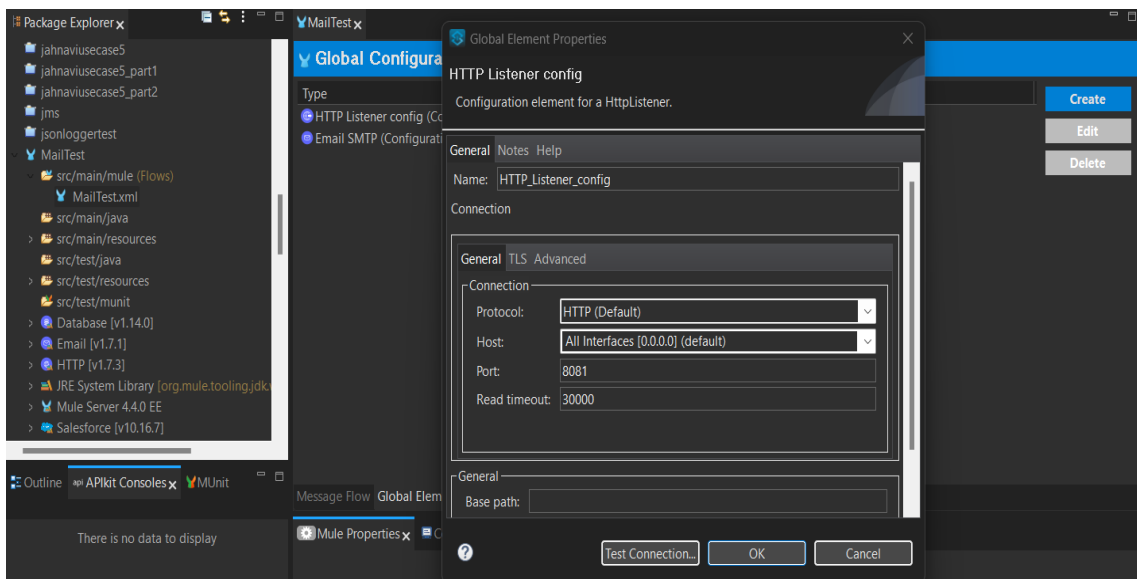
## Step 2:

- Now drag and drop a Flow and add an HTTP listener to it to create a source listener to get the mule event for the application.

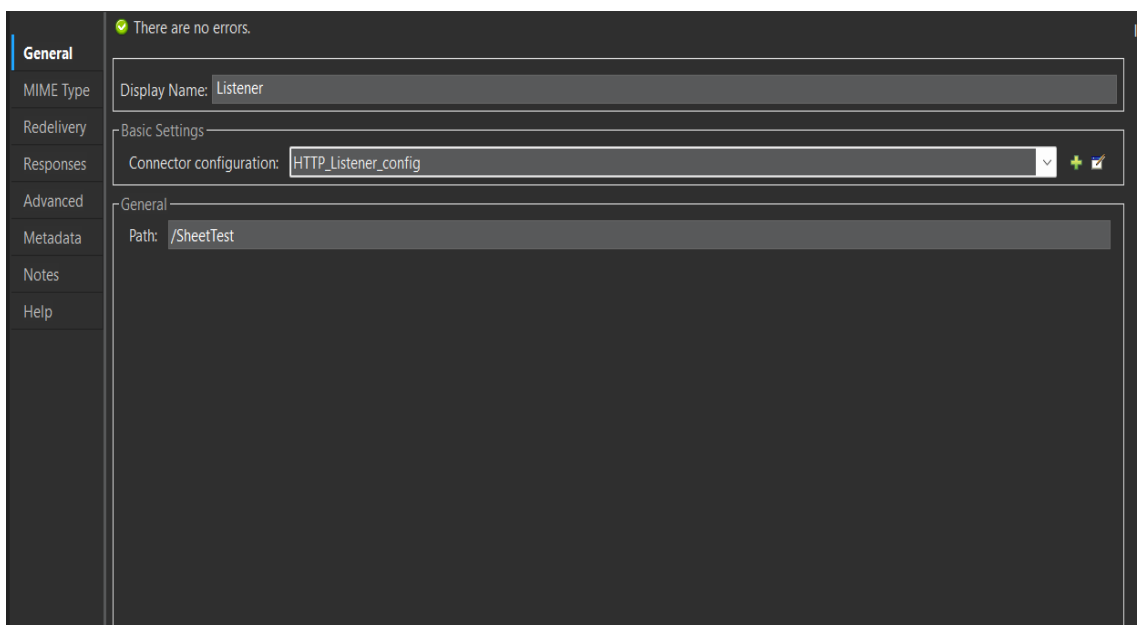


### Step 3:

- Now we need to configure the HTTP listener by providing the connector configuration and path.
- In connector configuration we need to provide the below details.
  - Protocol: HTTP or HTTPS
  - Host: All Interfaces[0.0.0.0](default)
  - Port:8081
- Rest we can leave as default.
- Then hit OK.

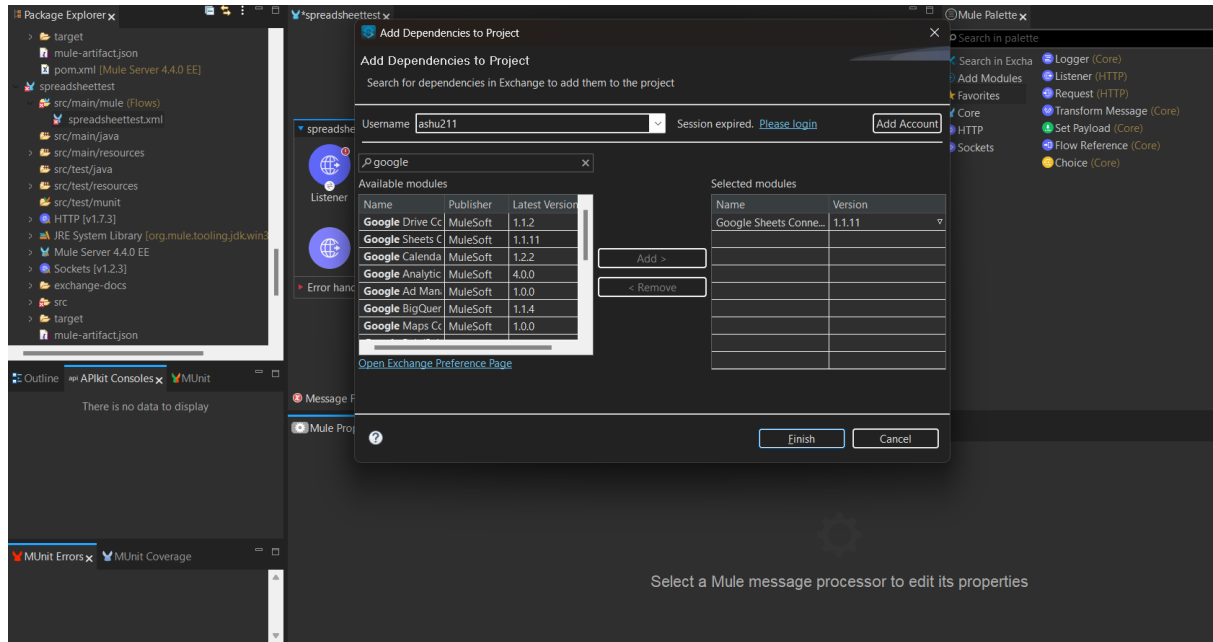


- For path we can define any resource path following “/”.(Example: /SheetTest)

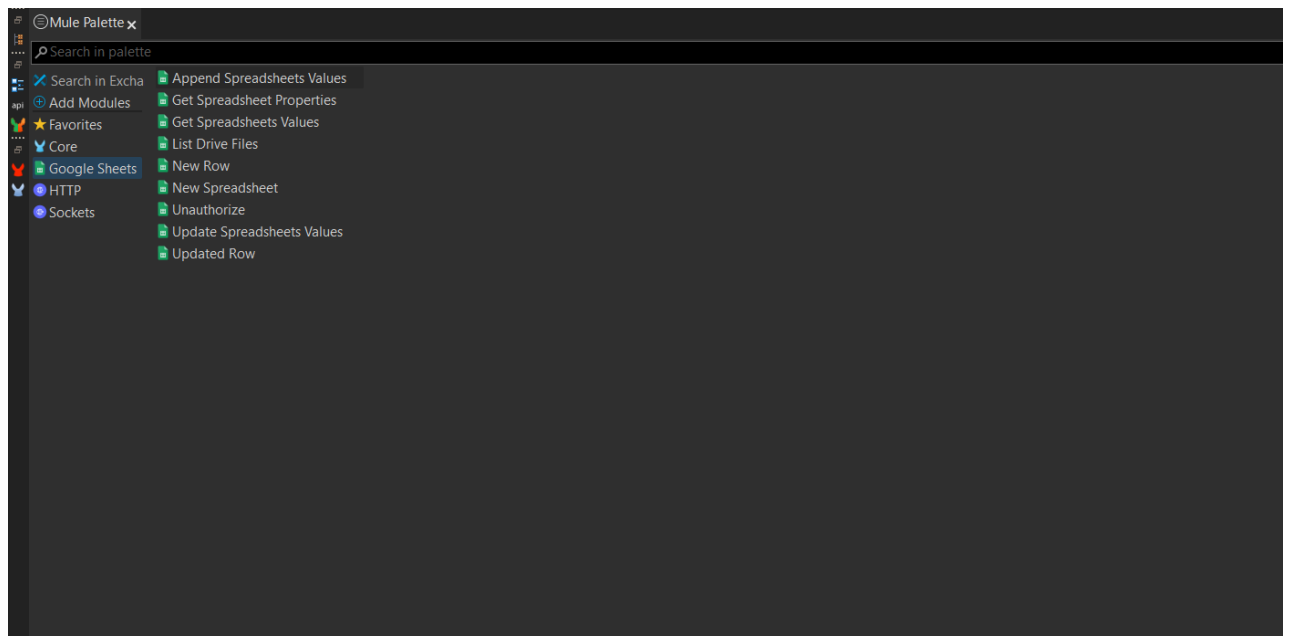


#### Step 4:

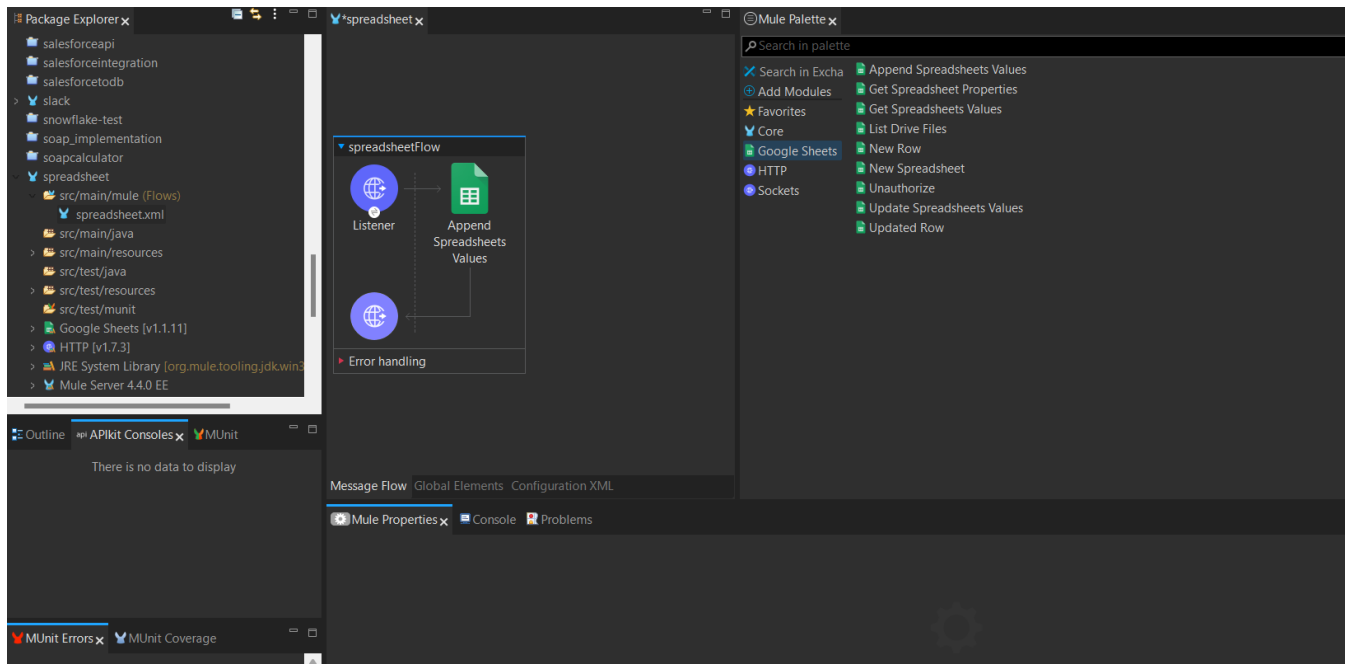
- Now we need to add a Google Sheet module to our mule palette. To do so we can search in the exchange about the Google Sheet module to add it to our anypoint studio. Once added we can see in the mule palette.



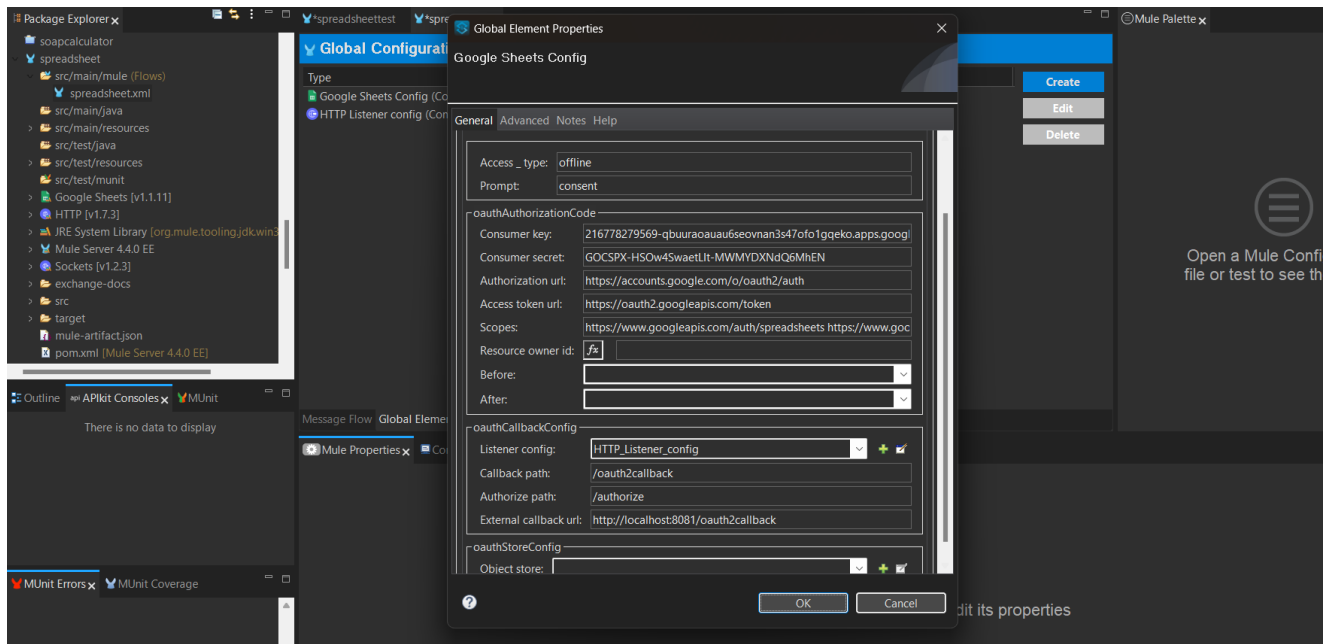
- Once we got the Google Sheets module we can see all the connectors to perform the required operation.



- We are using the Append Spreadsheet Values connector for the configuration for this particular scenario. We need to drag it to the mule flow.

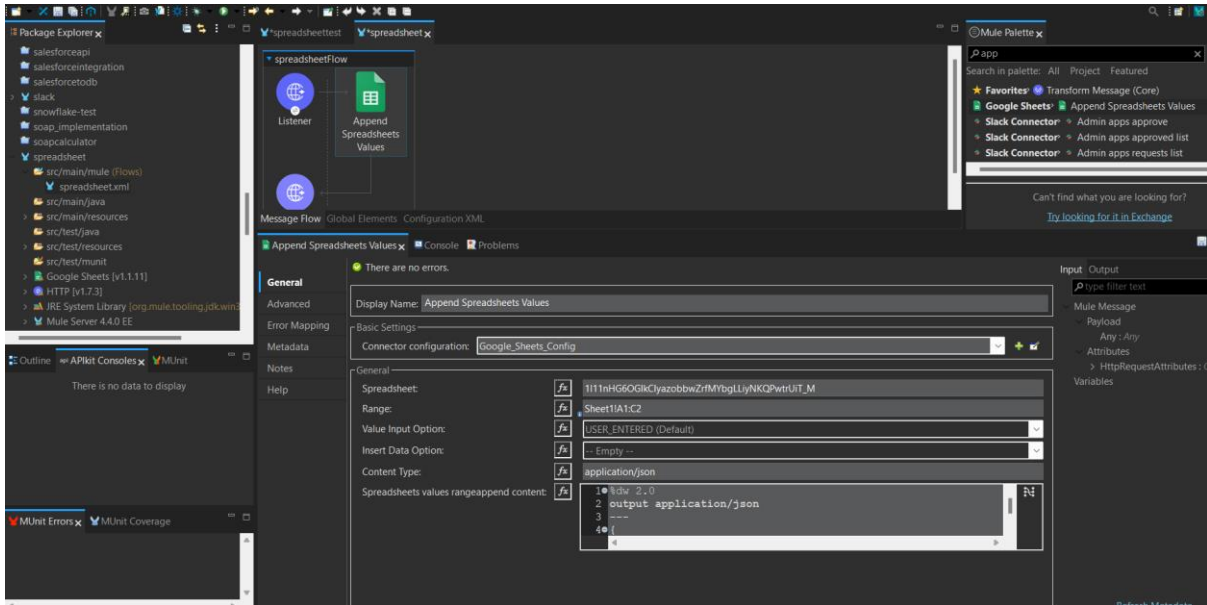


- In the Append Spreadsheet Values connector configuration section we need to hit the plus button.
- We need to provide the values for the following:
  - Access\_type : offline
  - Prompt: consent
  - Consumer Key: The Consumer ID provided in the Google API application
  - Consumer Secret: The Consumer Secret provided in the Google API application
  - Authorization URL: <https://accounts.google.com/o/oauth2/auth>
  - Access token URL: <https://oauth2.googleapis.com/token>
  - Scope: <https://www.googleapis.com/auth/spreadsheets>
  - Listener config: Choose the listener config of the HTTP listener
  - Callback path: /oauth2callback
  - Authorize path: /authorize
  - External Callback URI: http://localhost:8081/oauth2callback



## Step 6:

- Then we need to provide the details below to complete the setup for the Append Spreadsheet Values connector.
  - Spreadsheet: The Spreadsheet ID which we can get from the URL of the spreadsheet.
  - Range: Sheet1!A1:C2 (The range of the spreadsheet columns in which we want to append the data)
  - Value Input Option: Default
  - Insert Data Option: Empty
  - Content-Type: application/json



- Spreadsheet values range append contents: The payload that we want to store in a spreadsheet in the below format.

```
%dw 2.0
```

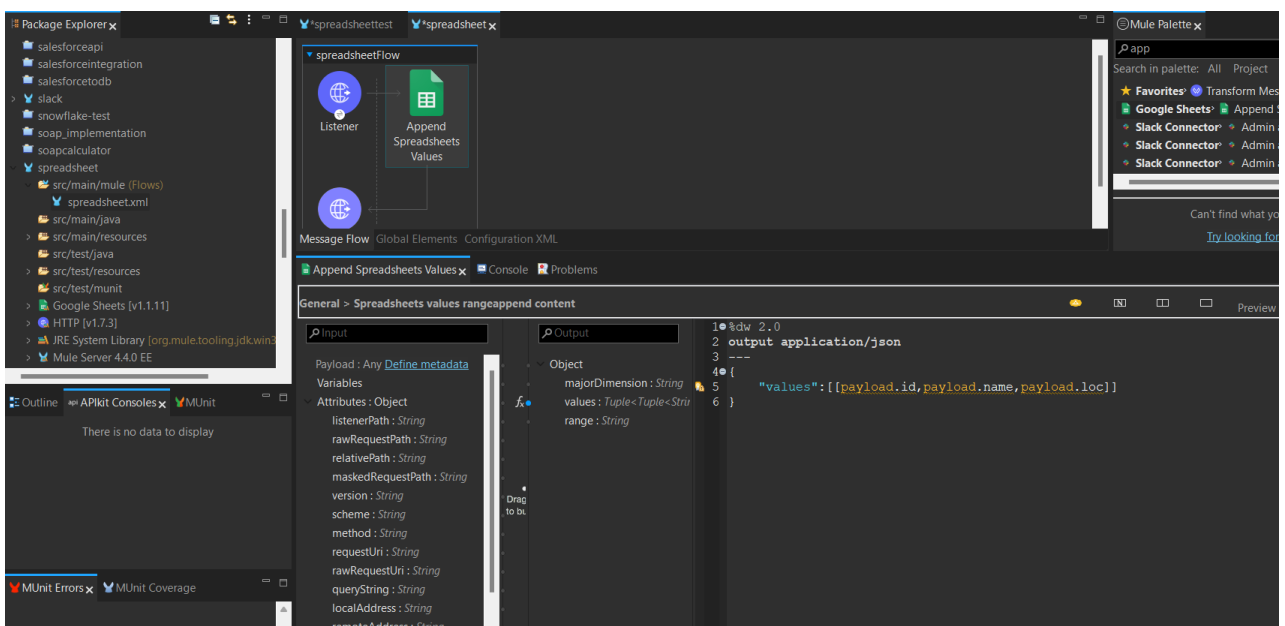
```
output application/json
```

```
---
```

```
{
```

```
  "values":[[payload.id,payload.name,payload.loc]]
```

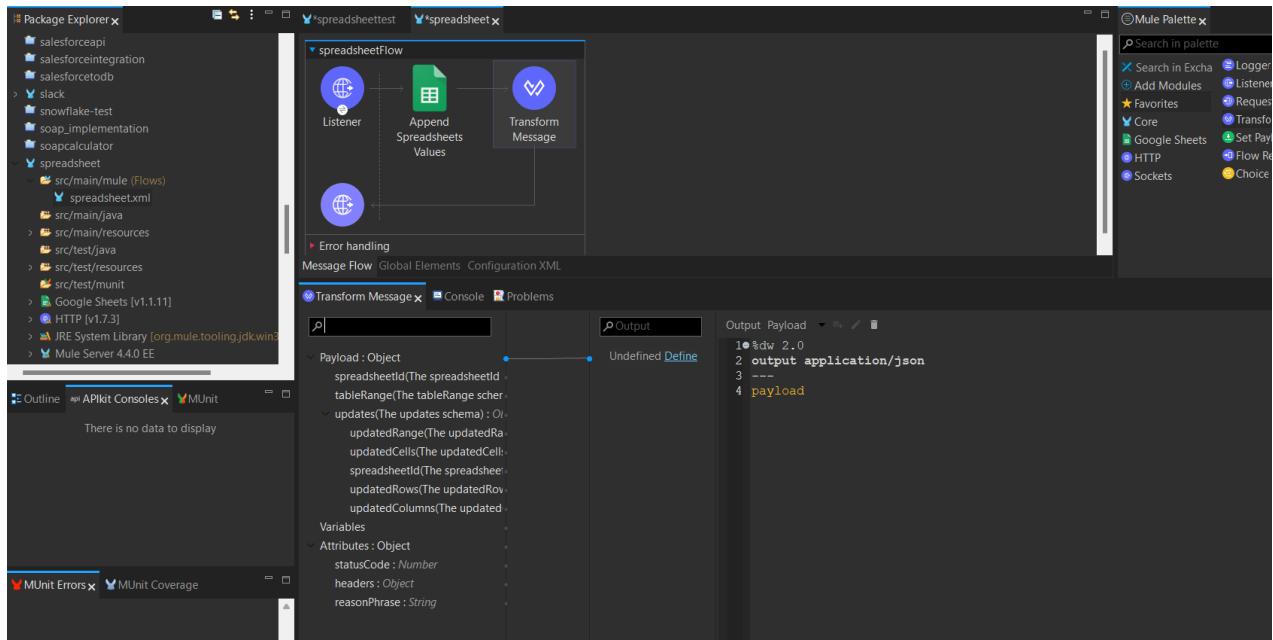
```
}
```





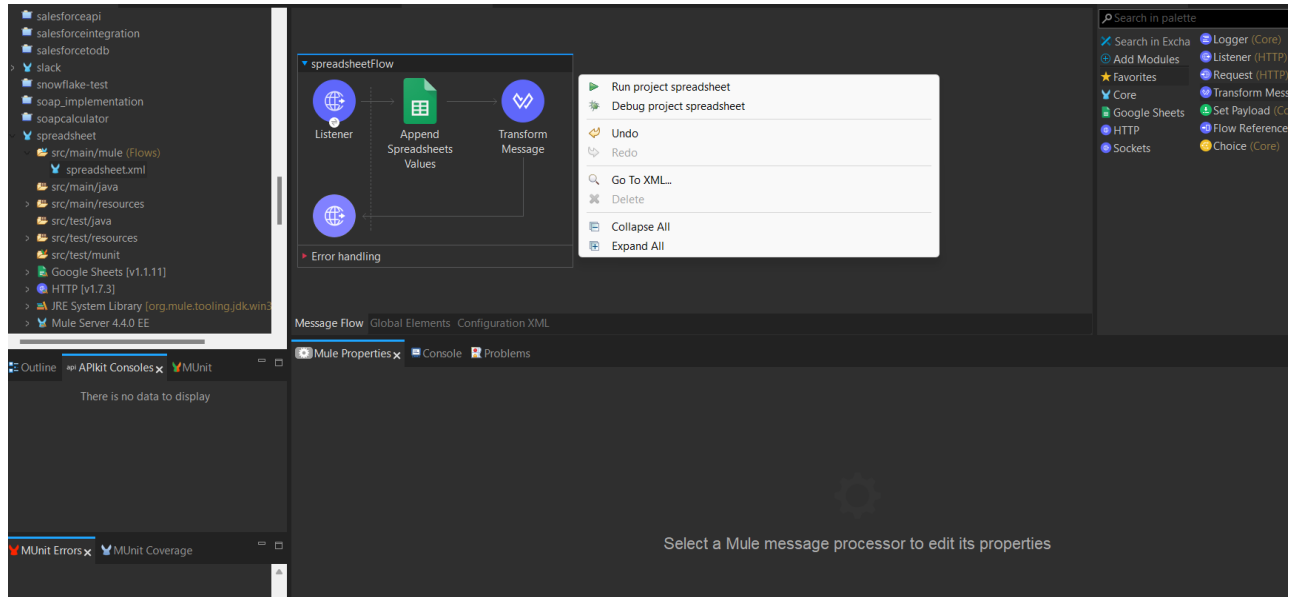
## Step 7:

- Now drag and drop a Transform Message shape to transform the output payload to JSON format by using the Dataweave  
%dw 2.0  
output application/json  
---  
payload

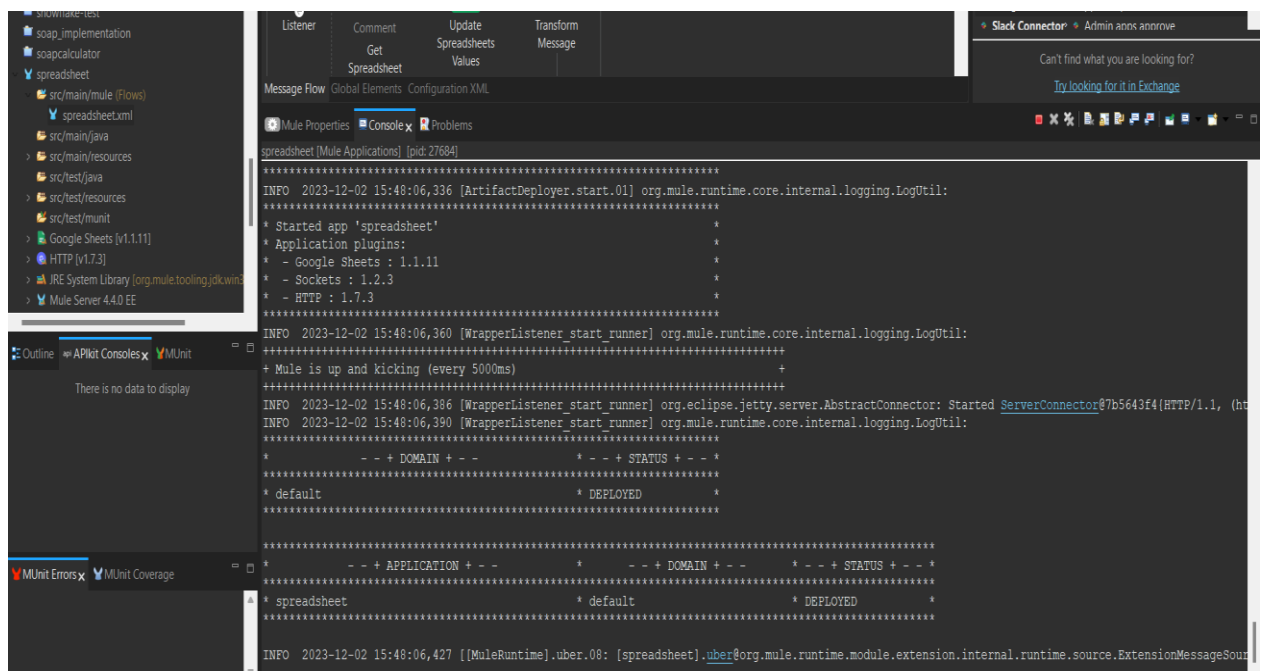


## Step 8:

- Now hit on Save and Run the project.



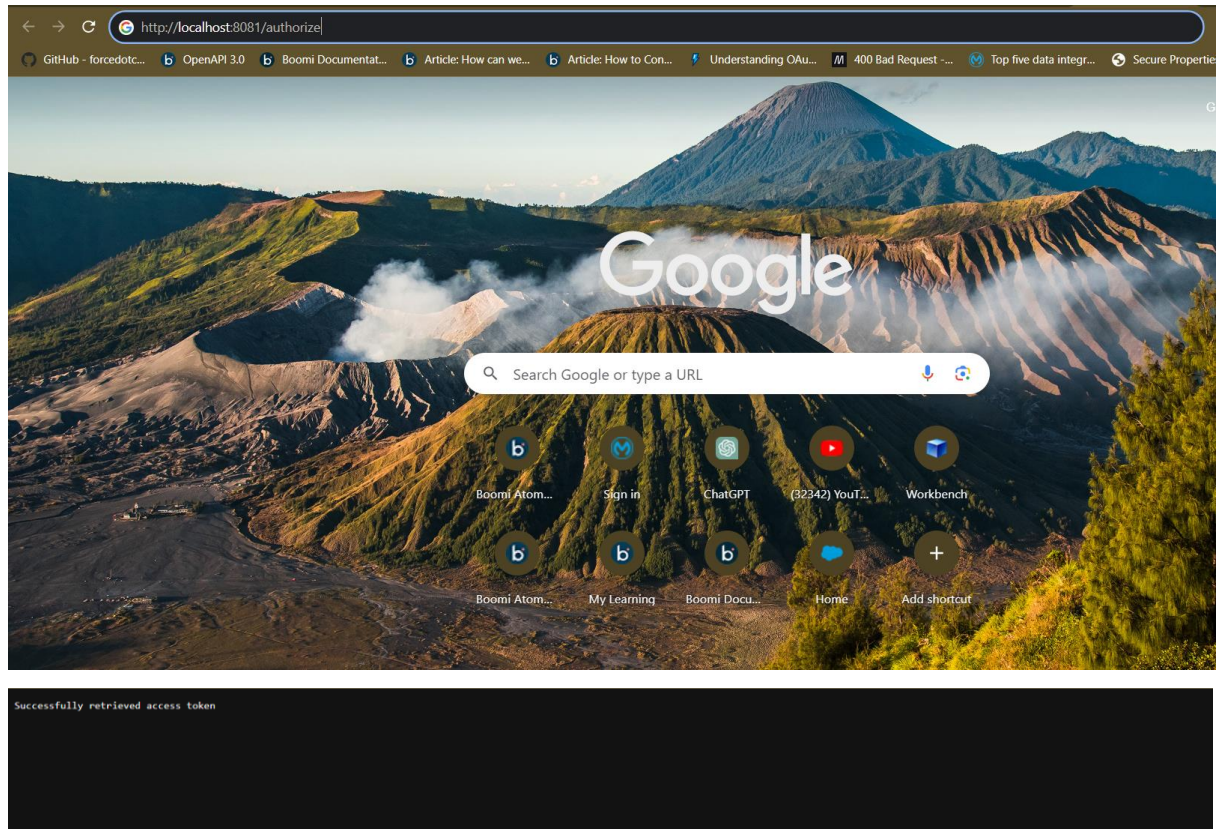
- In the console we can see the “Status” as “Deployed” to ensure our app is deployed and ready to be tested.



- Once the Application is up and running we can Test it by triggering the Mule app from Postman.

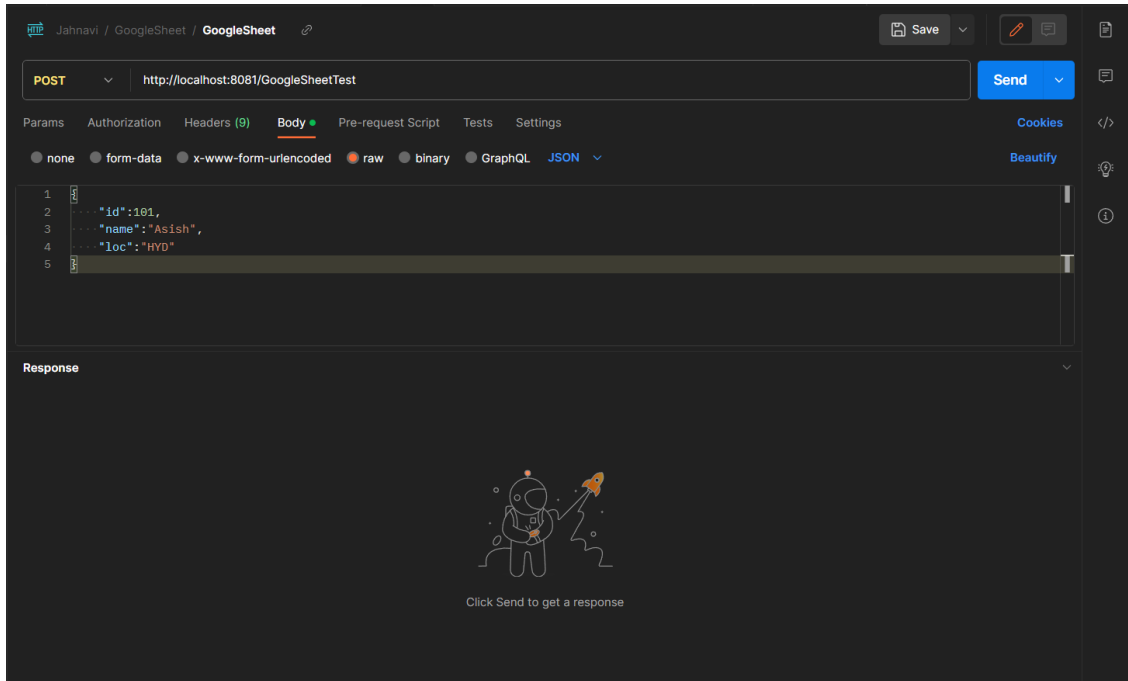
### **Step 10:**

- Now we need to go to any browser and hit <http://localhost:8081/authorize>.
- Now we need to log in with the Mail ID which we configured in the Google Spreadsheet API and provide the necessary permissions to get the Access token.

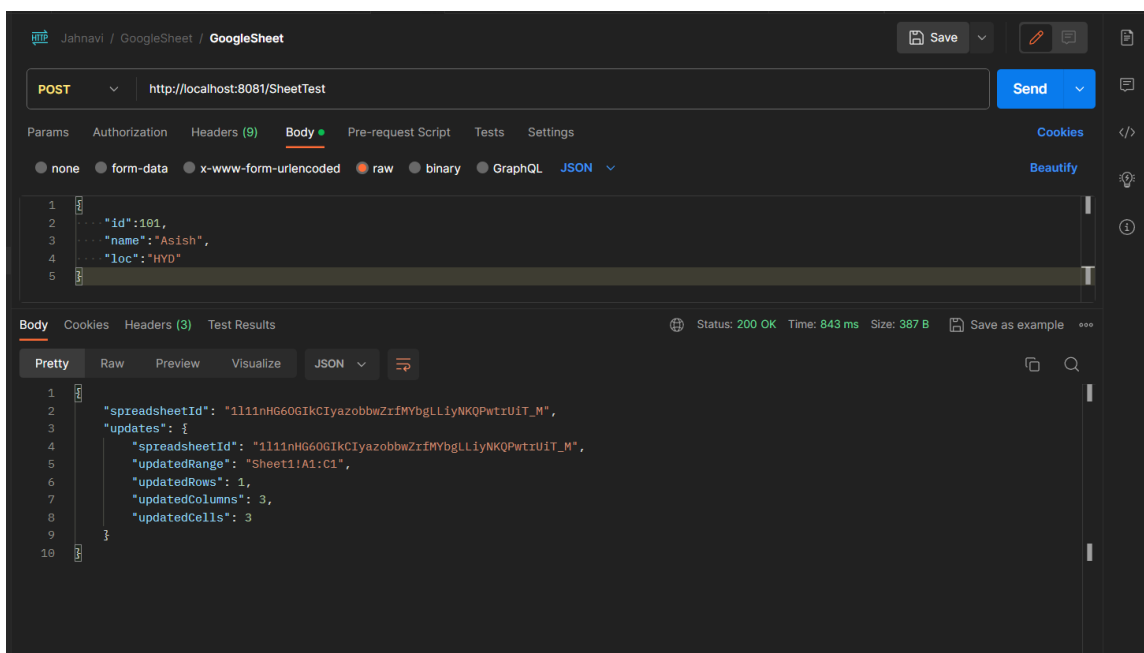


### **Step 11:**

- To trigger the Mule app we need to create a Request and we can hit the URL(<http://localhost:8081/GoogleSheetTest>) from Postman.
- We need to send a JSON body from the postman to the Mule Application to store the data in the Spreadsheet.



- We can see in the response the data is appended in the Spreadsheet.



- Now we can see in the console that the mule app got executed and we can check the Spreadsheet for the appended values.



The image shows a screenshot of a Google Sheets spreadsheet. The spreadsheet has columns labeled A through O and rows numbered 1 through 28. The data in row 1 is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	101	Asish	HYD												
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															





# TGH

Making Integrations Simpler

## TGH Software Solutions Pvt. Ltd.

[www.techygeekhub.com](http://www.techygeekhub.com)

At TGH, we specialize in driving digital transformation through seamless Integration Technologies.

Operating as an INTEGRATION FACTORY, we serve as a one-stop shop for all your integration needs. Our expert team is well-versed in enterprise software and legacy system integration, along with leading iPaaS technologies like Boomi, MuleSoft, Workato, OIC, and more.

We're committed to enhancing business processes and solving problems through our integration expertise.



### Email address

[connect@techygeekhub.com](mailto:connect@techygeekhub.com)



### Phone number

+ 011-40071137  
+ 91-8810610395



### Our offices

#### Noida Office

iThum  
Plot No -40, Tower A,  
Office No: 712,  
Sector-62, Noida,  
Uttar Pradesh, 201301

#### Hyderabad Office

Plot no: 6/3, 5th Floor,  
Techno Pearl Building,  
HUDA Techno Enclave,  
HITEC City, Hyderabad,  
Telangana 500081

