## **A Weekly Report Mr. Viraj Tank [EC034]**

## **[21ECUBG069] B.Tech. SEM. VIII In Partial Fulfillment of Requirement of Bachelor of Technology**

## **Degree of Electronics & Communication Course**

## **Submitted To**

**Faculty Supervisor**

**Dr. Narendra V. Chauhan**

## 

## **Department of Electronics & Communication Engineering**

## **Faculty of Technology,**

## **Dharmsinh Desai University, Nadiad-387001.**

## **(March 2025)**

**Task 1 :- Google Sheets Data Entry with Python and Google APIs**

**Objective :-**

The goal of this task is to make easy the process of adding data to Google Sheets by automating it with Python and Google APIs. This eliminates the need for manual data entry which reducing the chances of errors and saving time. This task focuses on creating a Python script that can connect to a Google Sheet, write data programmatically making it ideal for use cases like managing inventories.

**Scope :-**

The scope of this project involves automating data entry into Google Sheets using a Python-based solution. It includes setting up API access, authenticating through a service account, and developing a script to connect to a specific Google Sheet and append data programmatically. The project is designed to handle small to medium-scale datasets and focuses on tasks such as adding product details or similar structured information. It is limited to data entry and does not cover advanced operations like data analysis or visualization. Future enhancements could extend the script to handle bulk uploads, integrate scheduling, or process dynamic datasets from external sources.

**Tools and Technology Used :-**

1. Python:

Used for writing the automation script to interact with Google Sheets.

1. gspread Library:

A Python library for accessing and managing Google Sheets using the Google Sheets API.

1. oauth2client Library:

Used for handling OAuth 2.0 authentication, allowing secure access to Google services via a service account.

1. Google Sheets API:

Enables programmatic access to Google Sheets, allowing the script to read, write, and modify spreadsheet data.

1. Service Account:

A Google Cloud account used for authentication, providing secure access to the required Google Sheet.

1. JSON Key File:

A credentials file generated in Google Cloud, containing the service account details needed for authentication.

1. Google Drive API :

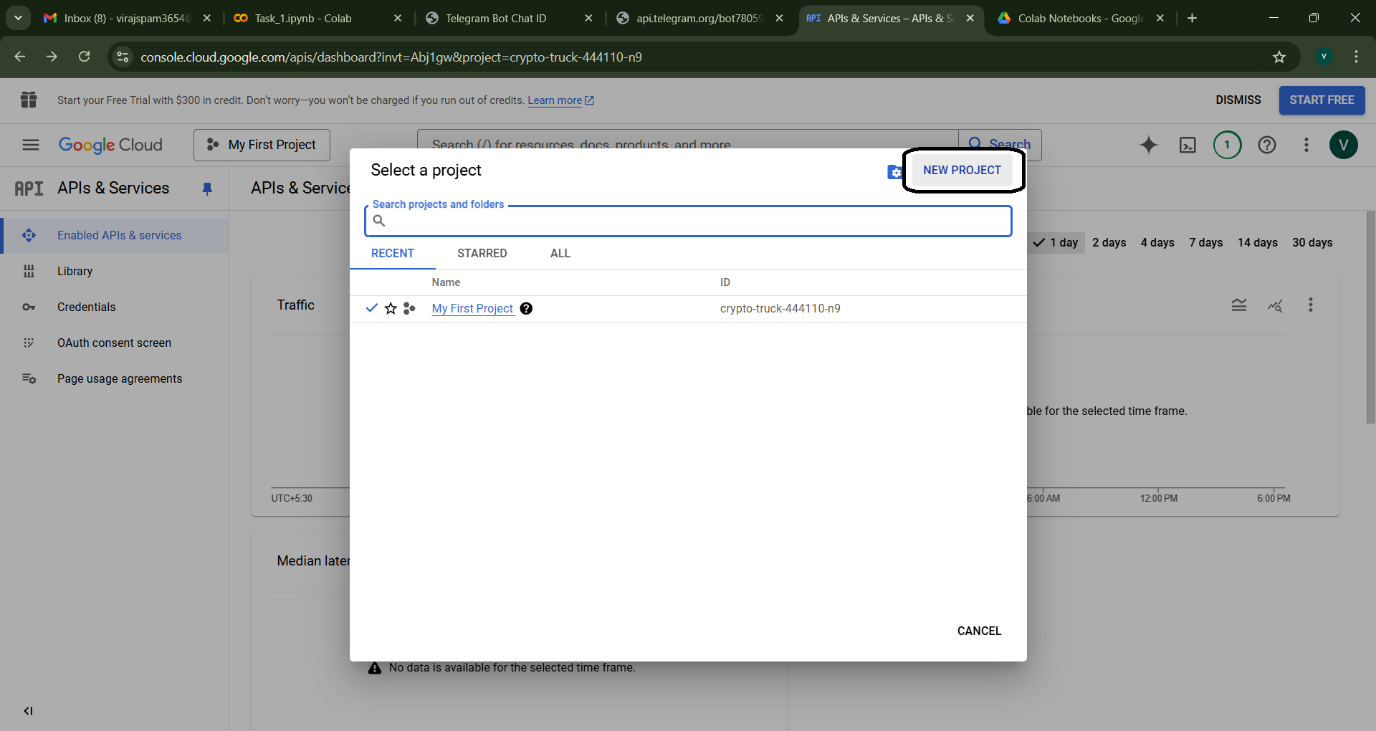
Allows the script to manage file permissions and ensure the service account has access to the target Google Sheet.

1. Google Colab :

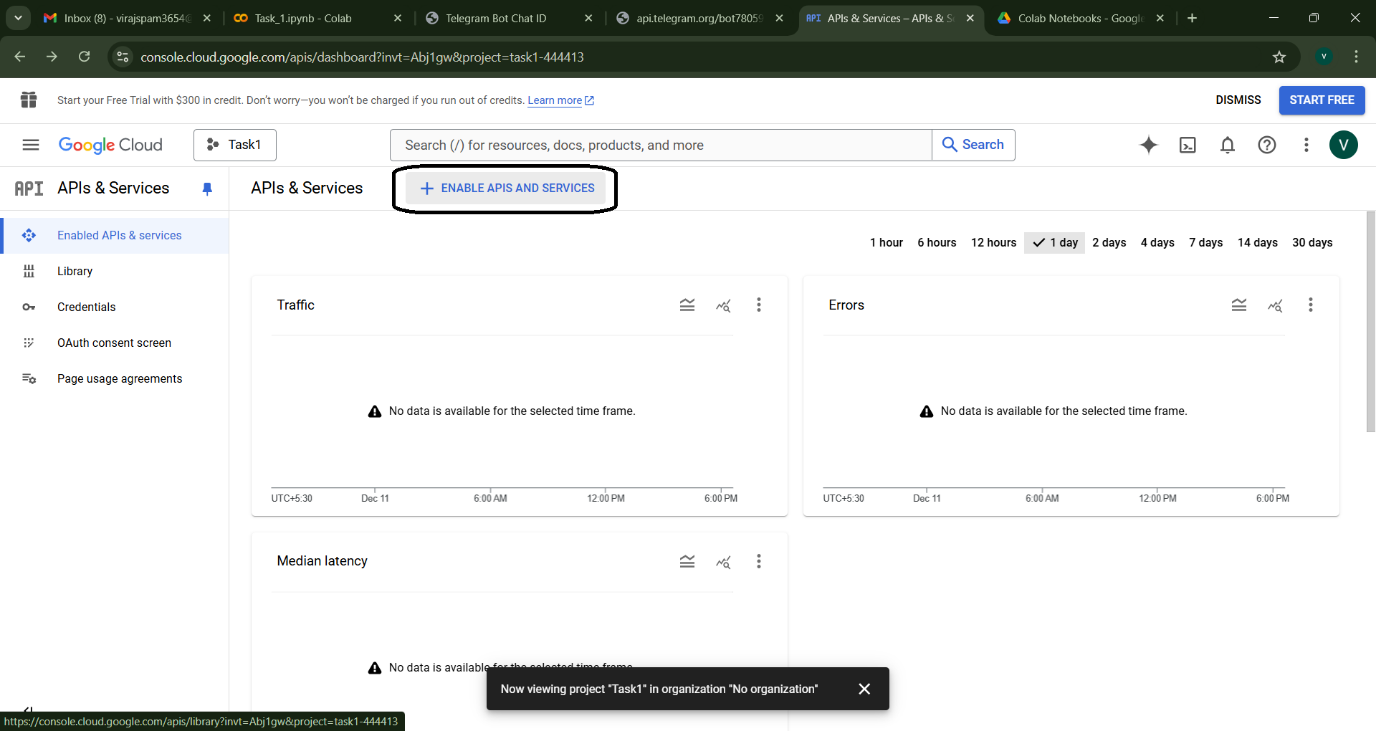
A cloud-based Python environment used for writing and testing the script.

**Setup Required :-**

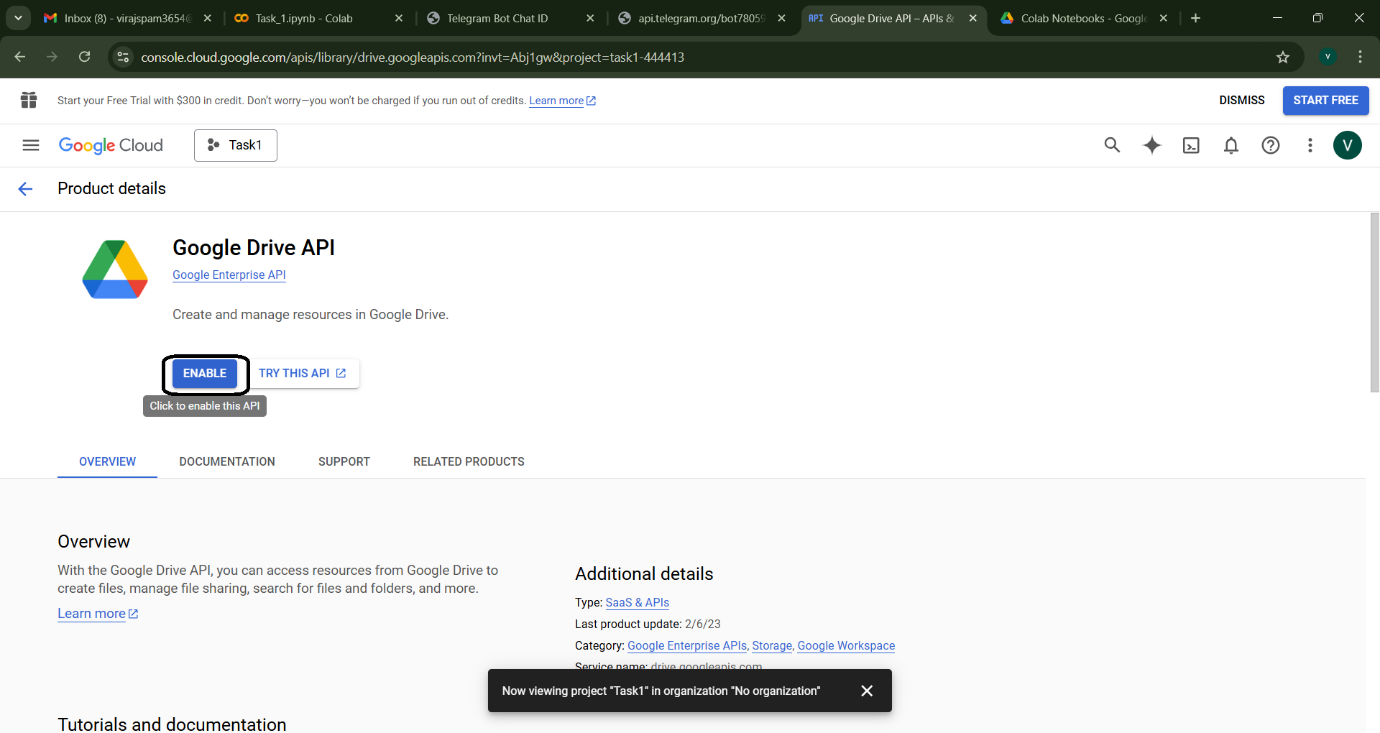
Step 1 :- Go to Google Cloud Console, click on New Project and provide Name of Project



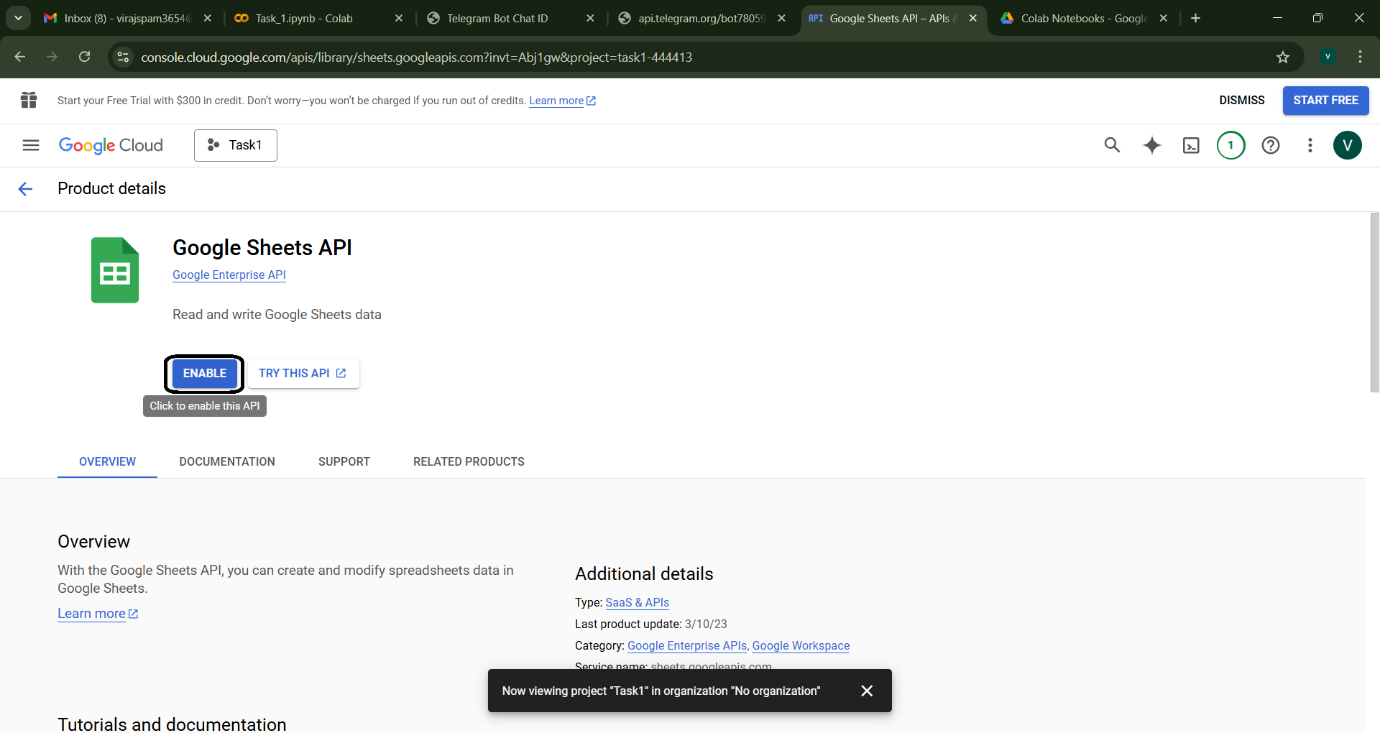
Step 2 :- Click on Enable APIs and Services



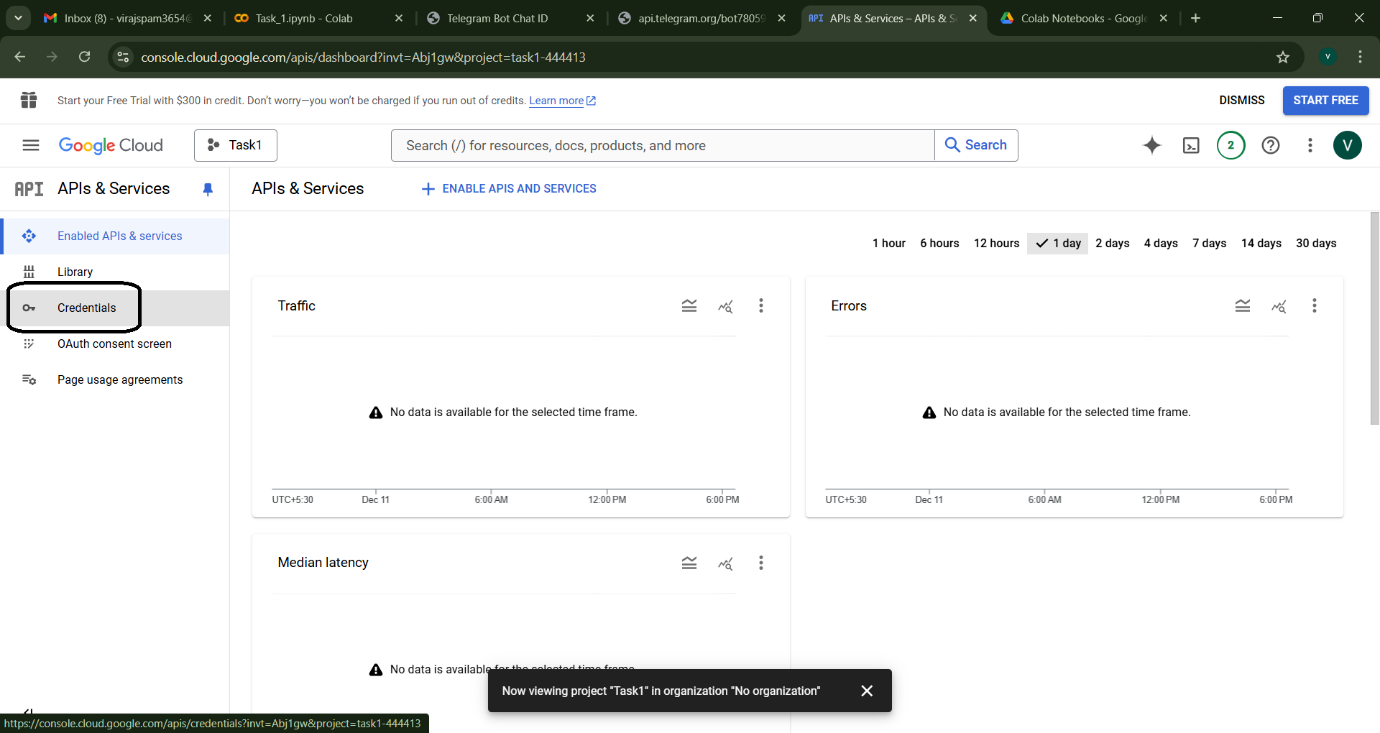
Step 3 :- Enable the Google Drive API



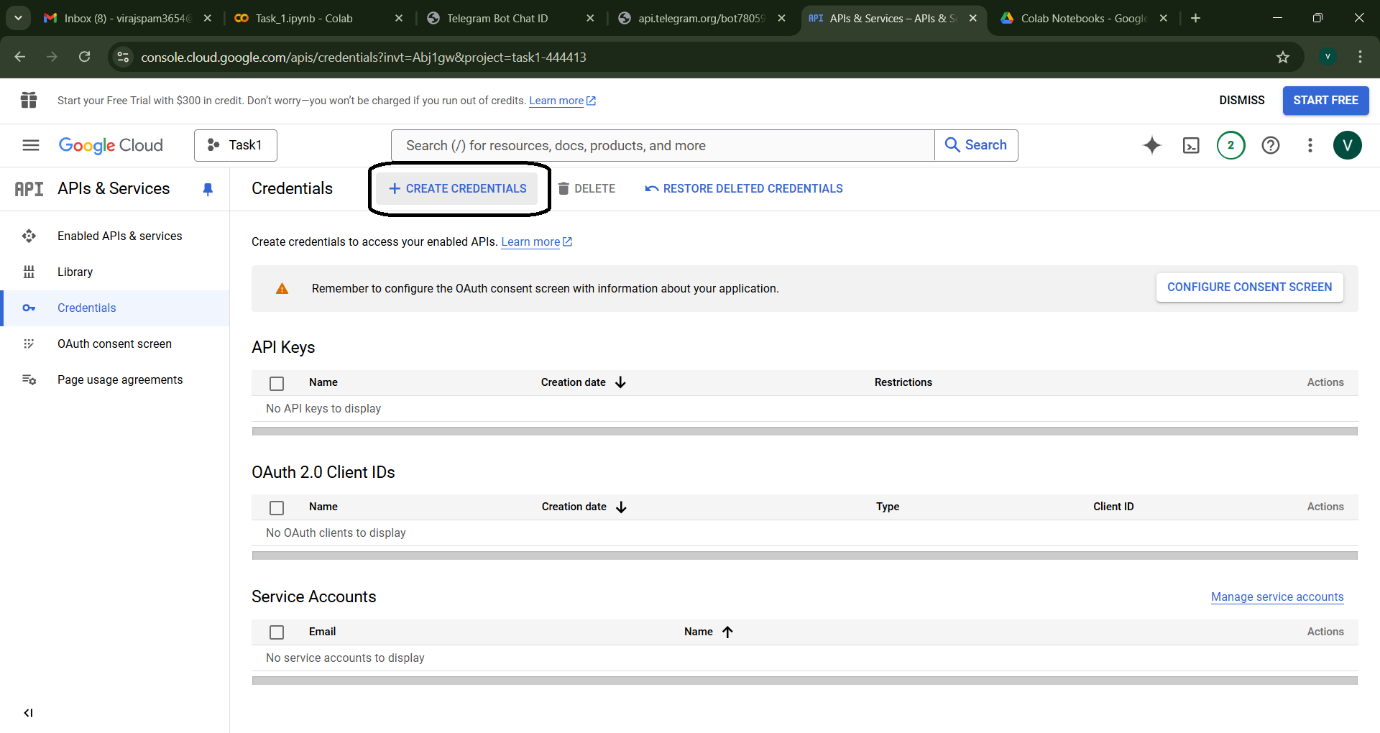
Step 4 :- Enable the Google Sheets API



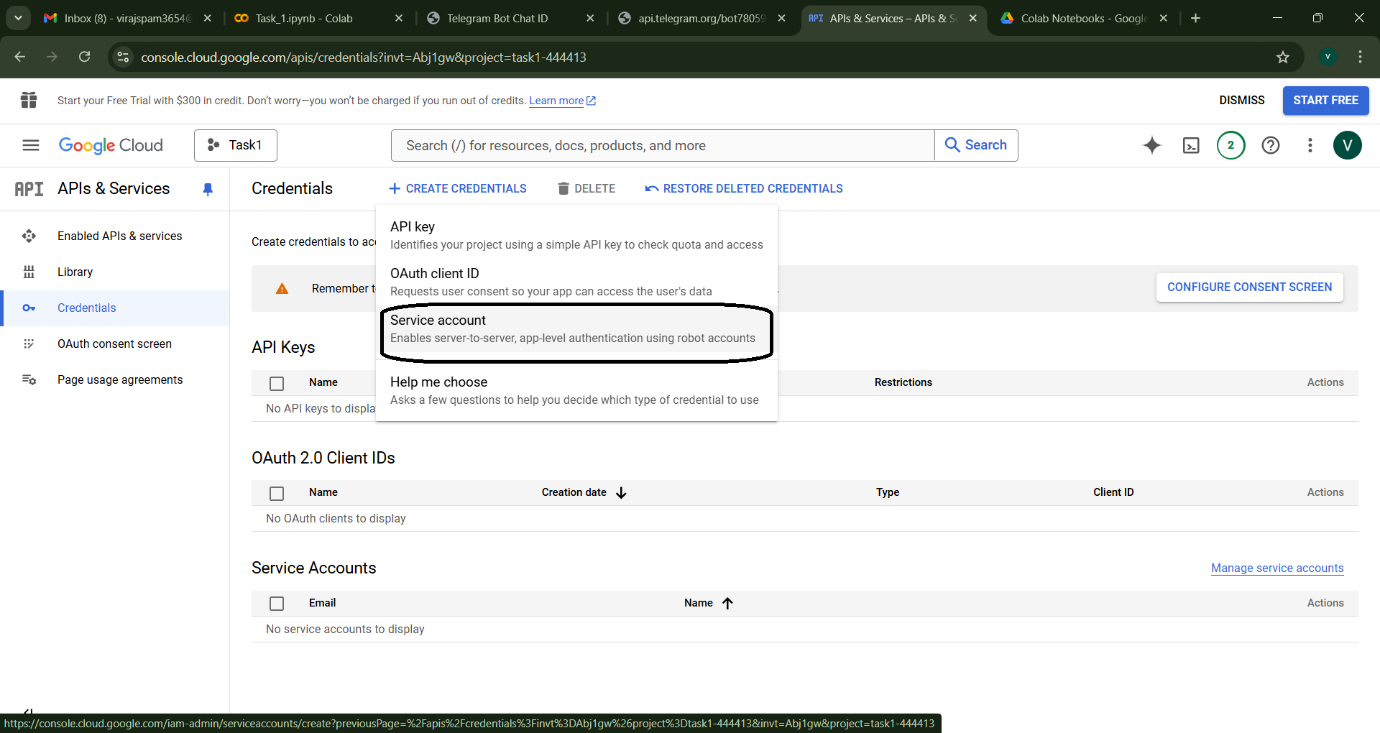
Step 5 :- Navigate to the APIs & Services > Credentials section in the Google Cloud Console.



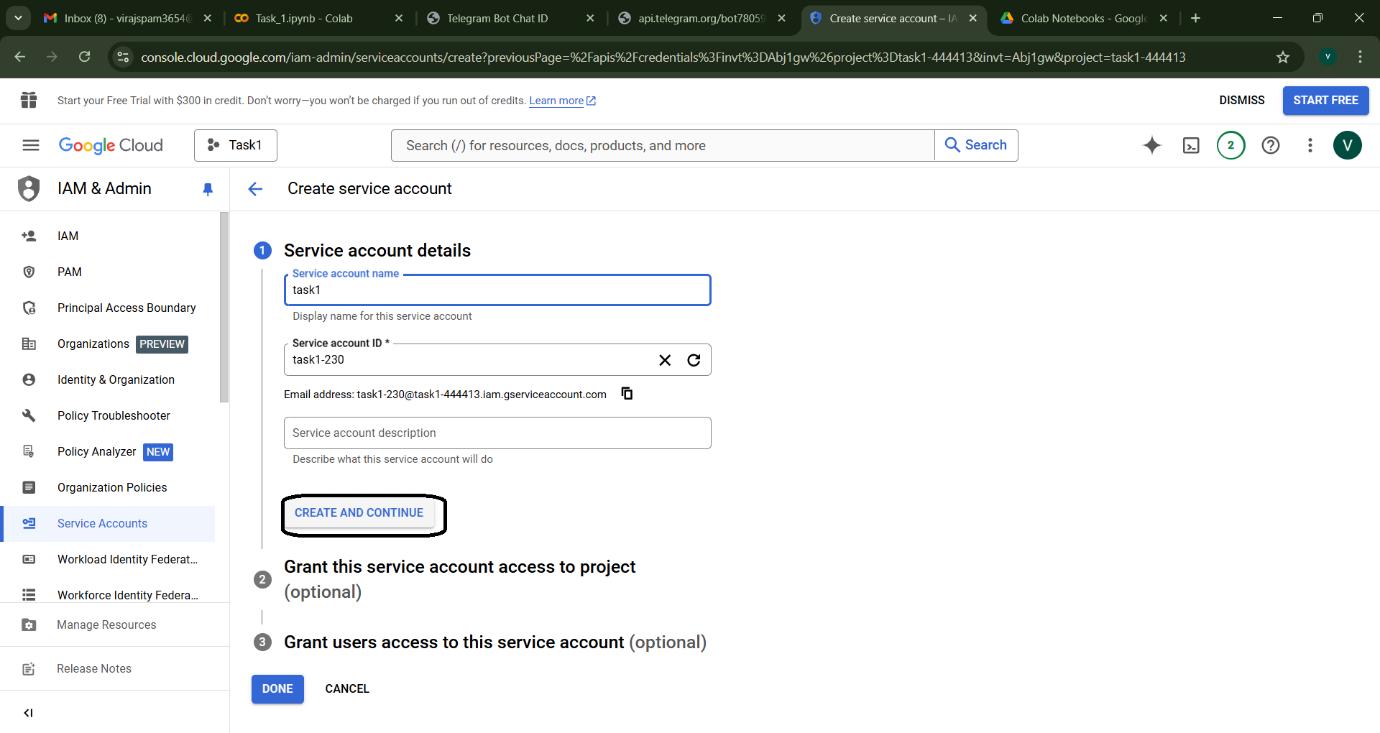
Step 6 :- Click on + Create Credentials



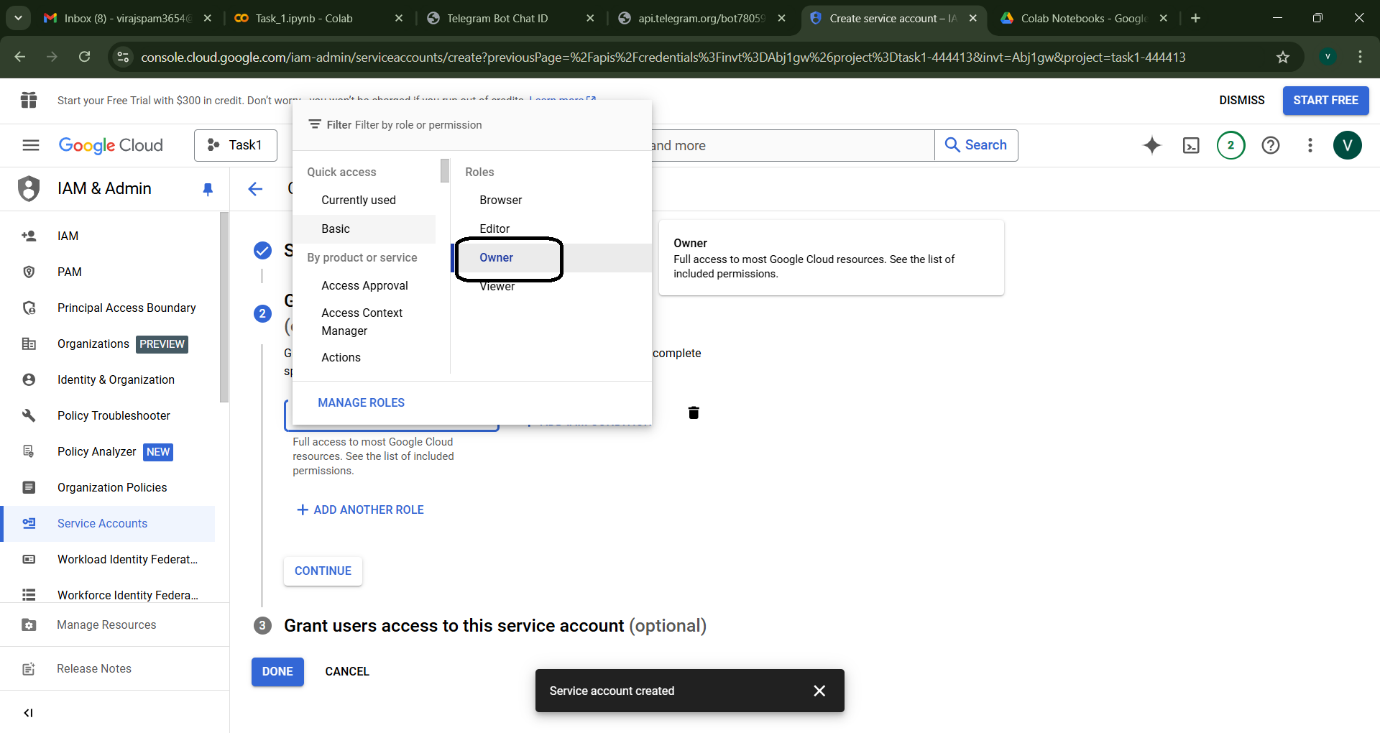
Step 7 :- And select Service Account



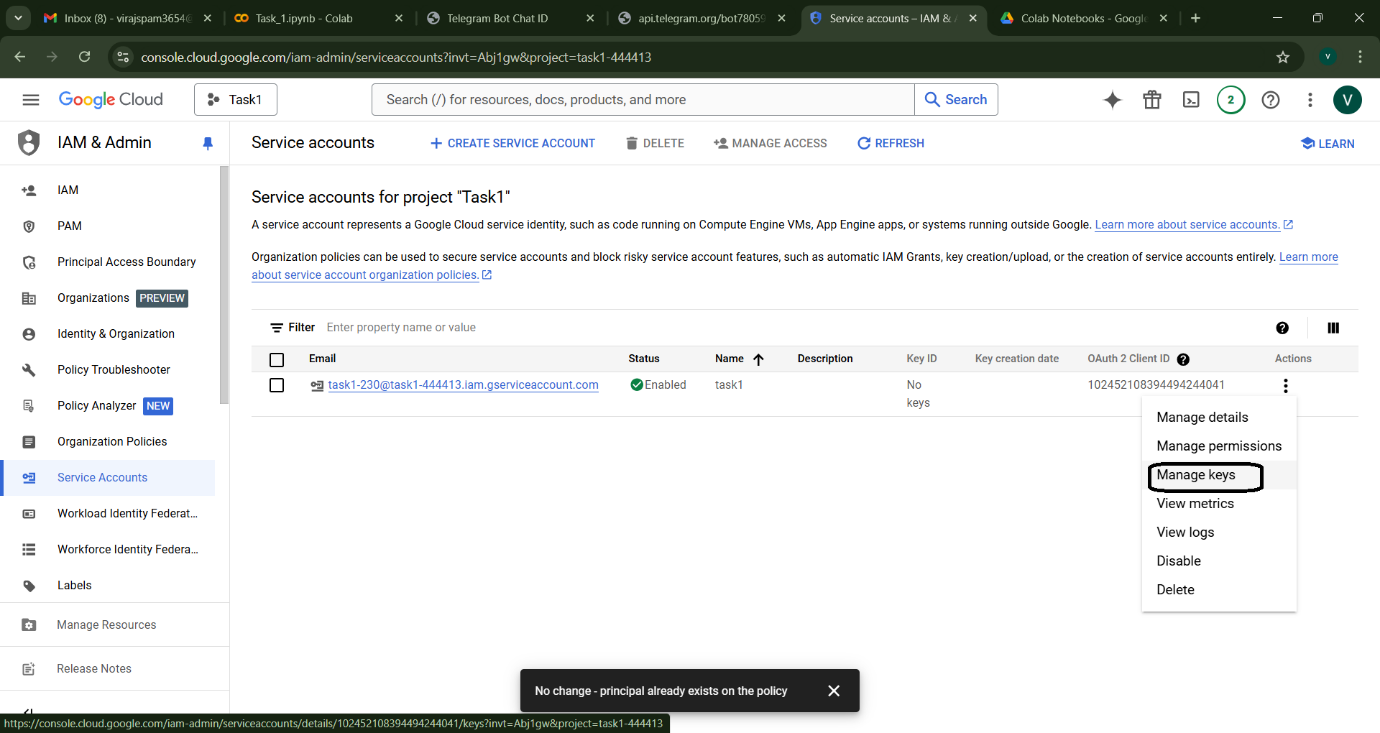
Step 8 :- Fill in the required details for the service account (name, description) and click Create.



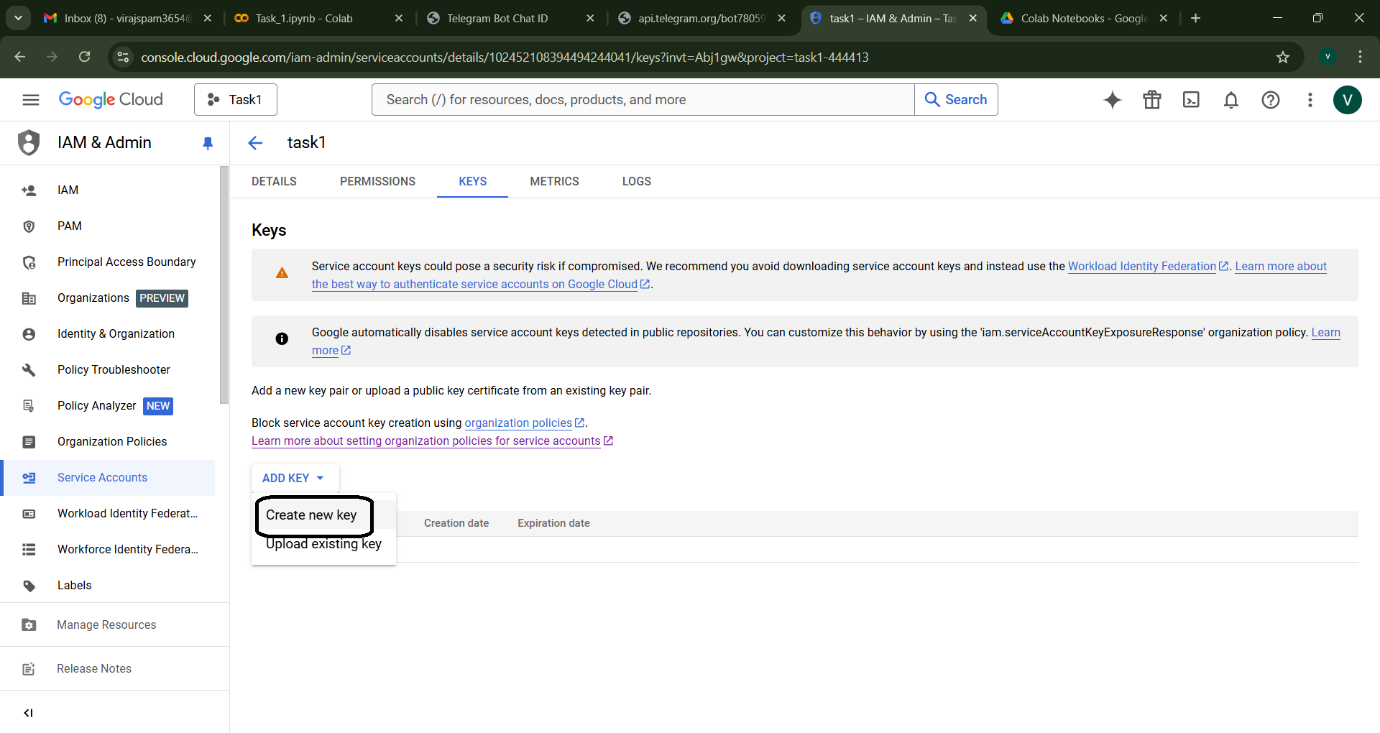
Step 9 :- Under the "Role" section, assign the service account a role, such as Editor or Owner. Complete the process, and in the last step, click Done.

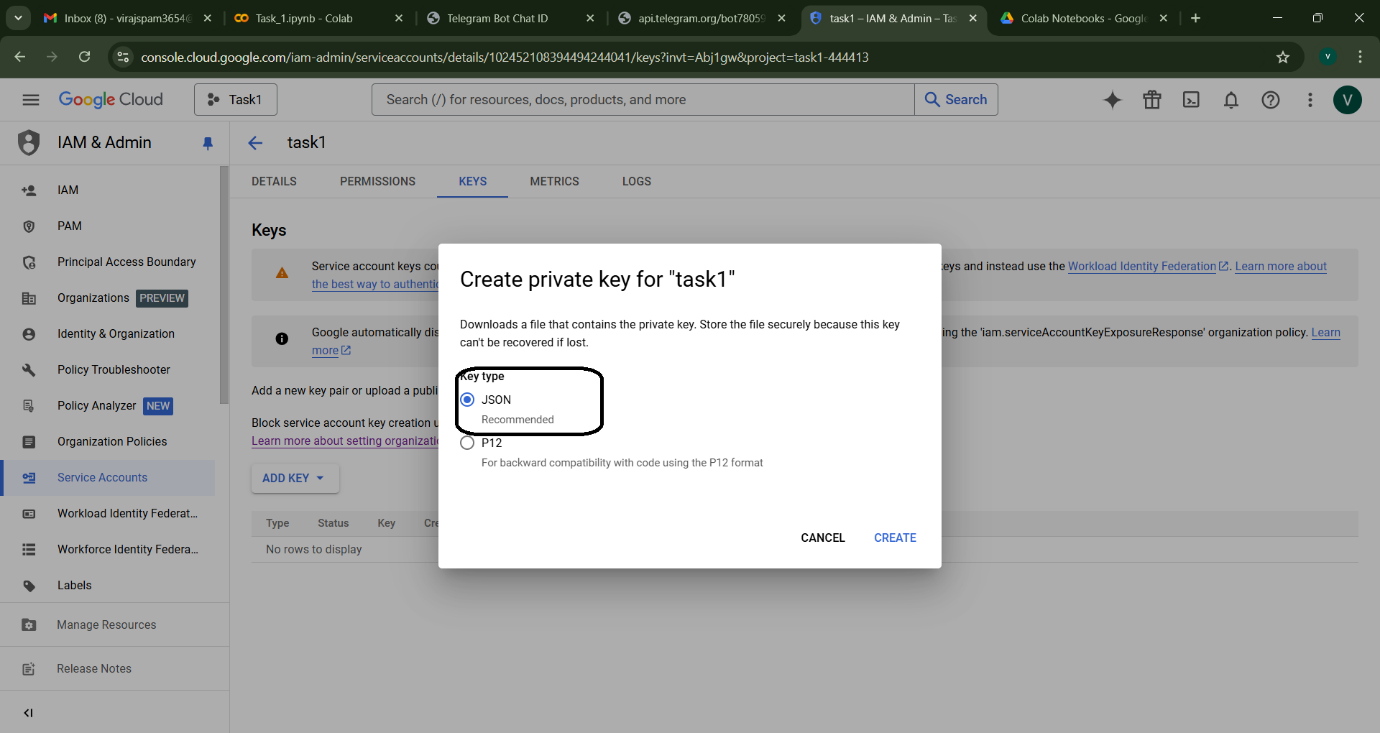


Step 10 :- After creating the service account, locate it in the "Credentials" tab and click Manage Keys.

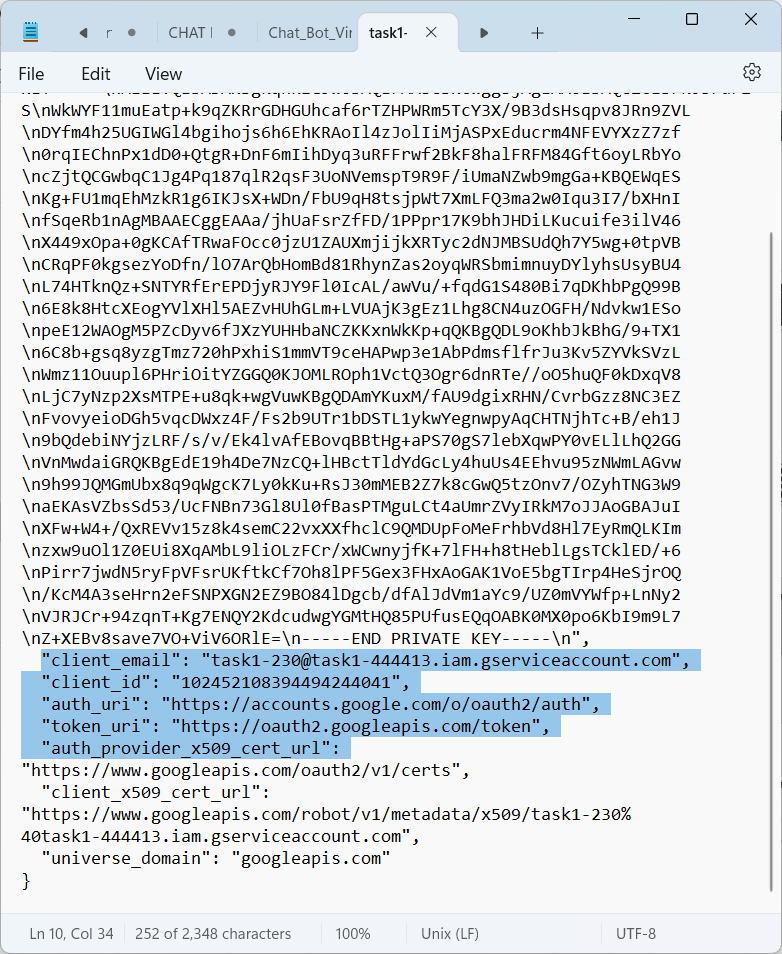


Step 11 :- Click Add Key > Create New Key, select the JSON option, and download the credentials file. And this file to Google Drive where code is present.

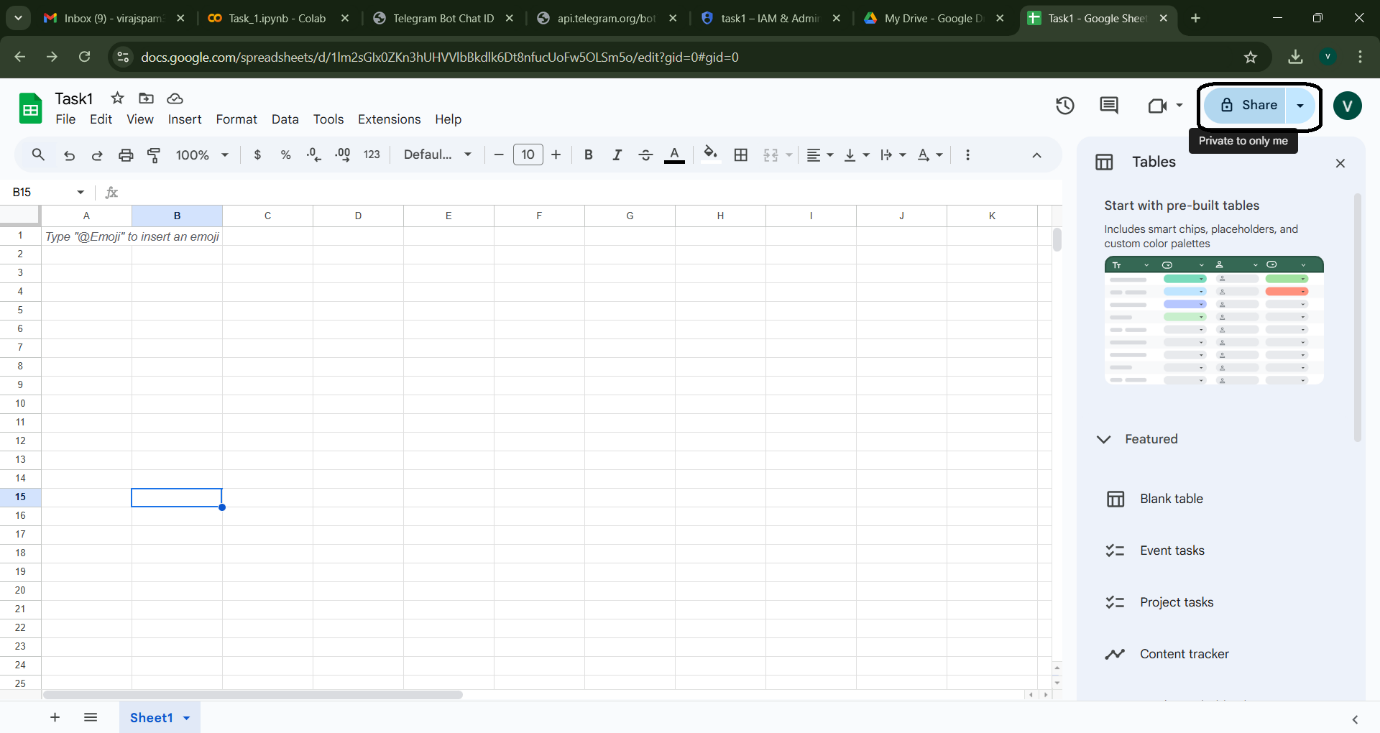




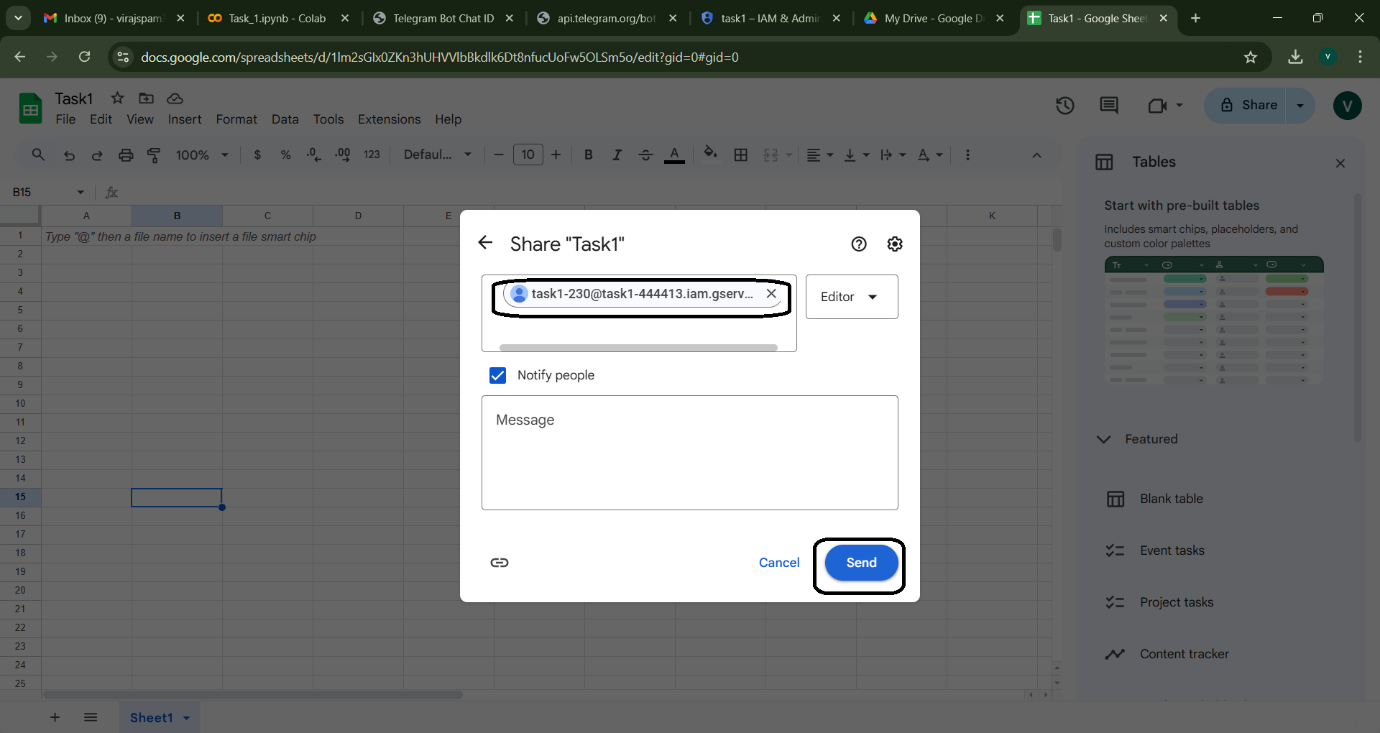
Step 12 :- Copy the email address of the service account (found in the ‘.json’ file under the client\_email field).



Step 13 :- Open the Google Sheet you want to use or create a new sheet.And Click on Share button.



Step 14 :- Share the Google Sheet with this email address by clicking Share in the Google Sheet and granting Editor access.



**Code :-**

*#Google Sheets Data Entry with Python and Google APIs*

*pip install gspread oauth2client*

*import gspread*

*from oauth2client.service\_account import ServiceAccountCredentials*

*def connect\_to\_google\_sheet(sheet\_name):*

*# Define the scope of the API*

*scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]*

*# Authenticate using the credentials JSON file*

*credentials = ServiceAccountCredentials.from\_json\_keyfile\_name("/content/drive/MyDrive/Colab Notebooks/task1.json", scope)*

*client = gspread.authorize(credentials)*

*# Open the Google Sheet*

*sheet = client.open(sheet\_name).sheet1*

*return sheet*

*def add\_product\_data(sheet, products):*

*# Add product data*

*for product, price in products.items():*

*sheet.append\_row([product, price])*

*print("Data added successfully!")*

*if \_\_name\_\_ == "\_\_main\_\_":*

*# Name of the Google Sheet*

*SHEET\_NAME = "Task1"*

*# Product data*

*product\_data = {*

*"Laptop": 50000,*

*"Smartphone": 25000,*

*"Headphones": 1800,*

*"Monitor": 3000,*

*}*

*# Connect to the Google Sheet*

*try:*

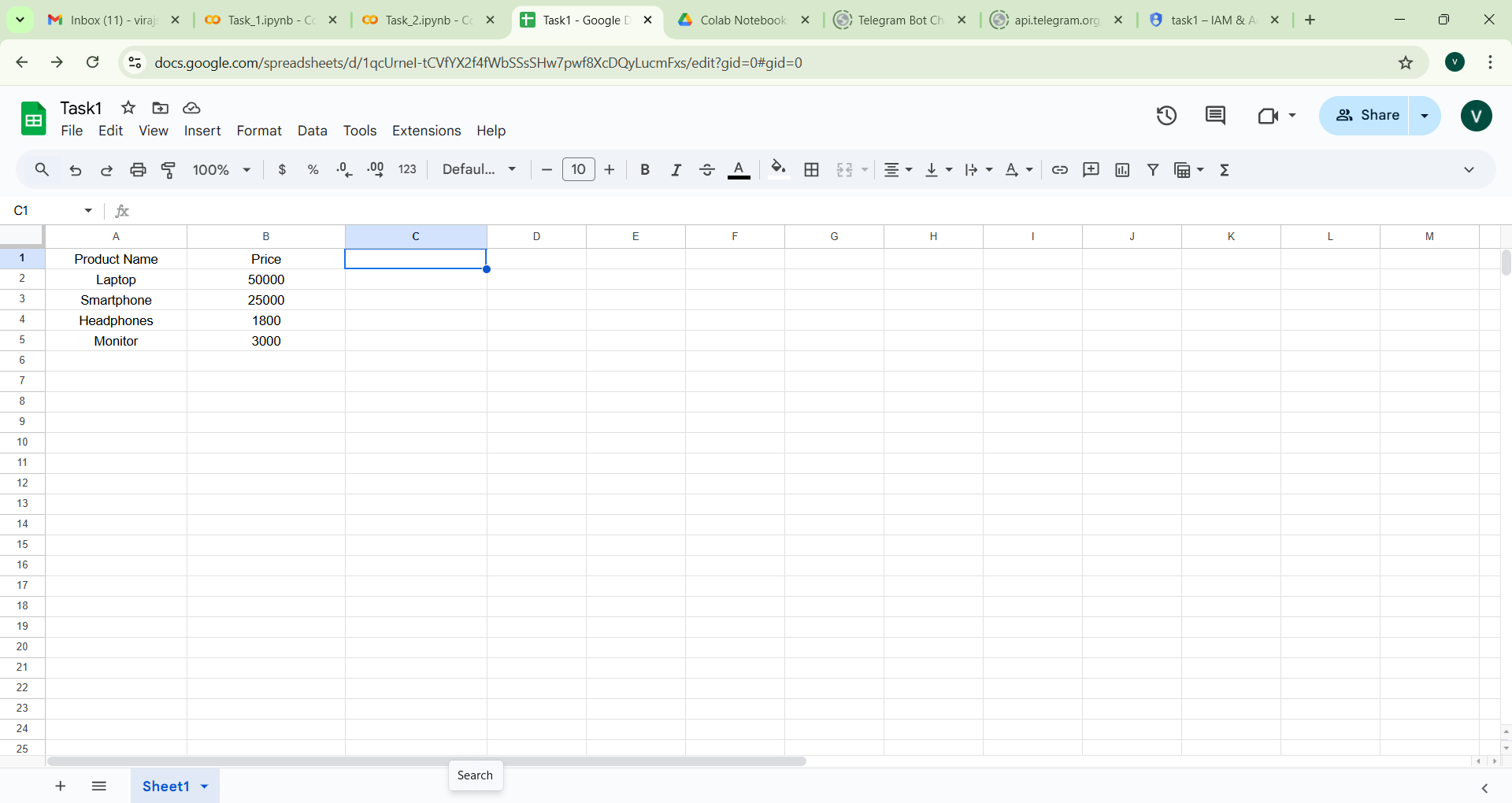
*sheet = connect\_to\_google\_sheet(SHEET\_NAME)*

*add\_product\_data(sheet, product\_data)*

*except Exception as e:*

*print("An error occurred:", e)*

**Output :-**



**Conclusion :-**

This task was a great learning experience, as it helped me understand how to use Python and Google APIs to automate data entry into Google Sheets. Overcoming challenges like API configuration and authentication taught me valuable skills. The final solution simplifies repetitive tasks, making data entry faster and more accurate. It also highlighted how automation can save time and improve productivity in real-world scenarios.

**Task 2 :- Automating Product Data Entry via Telegram and Google Sheets**

**Objective :-**

The objective of this task is to develop an automated system for efficiently recording product data into a Google Sheet using a Telegram bot. This system aims to simplify the data entry process by allowing users to send product details directly through a Telegram chat, eliminating the need for manual data entry. By the integration of Google Sheets API for backend storage and Telegram Bot API for user interaction, the project provides a seamless, user-friendly, and accessible interface for data input. Furthermore, the bot is designed to validate and process the data accurately, ensuring reliability. This automation enhances productivity and demonstrates the practical application of integrating messaging platforms with cloud-based tools

**Scope :-**

This project aims to create an automated system that simplifies recording product details into Google Sheets using a Telegram bot. It is designed for individuals, small businesses, and inventory managers who want a quick and easy way to manage product information without manual effort. Users can interact with the bot via Telegram by sending product details like name and price in a specific format. The bot checks the data to ensure it’s in the correct format and then saves it directly to a Google Sheet. By combining the convenience of Telegram with the functionality of Google Sheets, this project offers a practical, user-friendly solution for data entry tasks.

**Tools and Technology Used :-**

1. Python:

Used for writing the automation script to interact with Google Sheets.

1. gspread Library:

A Python library for accessing and managing Google Sheets using the Google Sheets API.

1. python-telegram-bot:

Simplifies the development of the Telegram bot by providing tools to handle messages, commands, and other bot-related functionality

1. nest\_asyncio:

Ensures compatibility with asynchronous tasks in environments like Jupyter Notebook or Google Colab.

1. oauth2client Library:

Used for handling OAuth 2.0 authentication, allowing secure access to Google services via a service account.

1. Google Sheets API:

Enables programmatic access to Google Sheets, allowing the script to read, write, and modify spreadsheet data.

1. Service Account:

A Google Cloud account used for authentication, providing secure access to the required Google Sheet.

1. JSON Key File:

A credentials file generated in Google Cloud, containing the service account details needed for authentication.

1. Google Drive API:

Allows the script to manage file permissions and ensure the service account has access to the target Google Sheet.

1. Google Colab:

A cloud-based Python environment used for writing and testing the script.

1. Telegram Bot API:

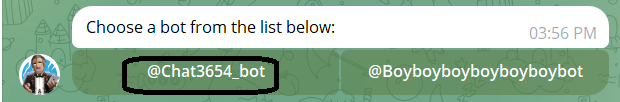
Provides the interface for creating and managing the Telegram bot, allowing it to interact with users efficiently.

**Setup Required :-**

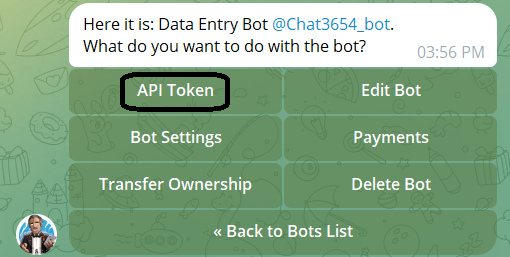
All Previous Steps remain same in addition to following steps.

Step 15 :- Create a Bot in telegram using BotFather.

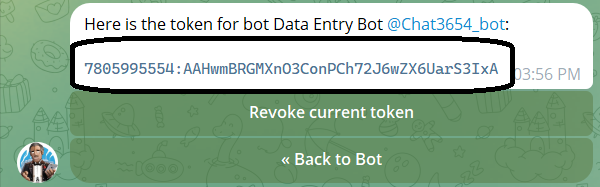
Step 16 :- Now type /mybots and select the username of bot used.



Step 17 :- Now select API Token to get bots API Token.



Step 18 :- Copy the API Token provided this will be used further to configure the bot in python script.



**Code :-**

*# Automating Product Data Entry via Telegram and Google Sheets*

*pip install gspread oauth2client python-telegram-bot nest\_asyncio*

*import asyncio*

*import nest\_asyncio*

*from telegram import Update*

*from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackContext*

*import gspread*

*from oauth2client.service\_account import ServiceAccountCredentials*

*nest\_asyncio.apply()*

*# Google Sheets Authentication*

*def connect\_to\_google\_sheet(sheet\_name):*

*# Defining the scope of the API*

*scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]*

*credentials = ServiceAccountCredentials.from\_json\_keyfile\_name("/content/drive/MyDrive/Colab Notebooks/task1.json", scope)*

*client = gspread.authorize(credentials)*

*sheet = client.open(sheet\_name).sheet1*

*return sheet*

*# Telegram Bot Handlers*

*async def start(update: Update, context: CallbackContext):*

*await update.message.reply\_text("Welcome! Send me product details in the format: Product Name, Price")*

*async def add\_data(update: Update, context: CallbackContext):*

*# Extract message content*

*message = update.message.text*

*try:*

*# Split the message to extract product and price*

*product, price = map(str.strip, message.split(","))*

*price = float(price)*

*# Connect to Google Sheet*

*sheet = connect\_to\_google\_sheet("Task1")*

*# Add the product and price to the sheet*

*sheet.append\_row([product, price])*

*await update.message.reply\_text(f"Added: {product} - Rs {price:.2f}")*

*except ValueError:*

*await update.message.reply\_text("Invalid format! Please send data in the format: Product Name, Price")*

*except Exception as e:*

*await update.message.reply\_text(f"An error occurred: {e}")*

*async def main():*

*# Telegram bot token*

*BOT\_TOKEN = "7805995554:AAHwmBRGMXnO3ConPCh72J6wZX6UarS3IxA"*

*# Initialize the bot application*

*application = Application.builder().token(BOT\_TOKEN).build()*

*# Add handlers*

*application.add\_handler(CommandHandler("start", start))*

*application.add\_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, add\_data))*

*# Start the bot*

*print("Bot is running...")*

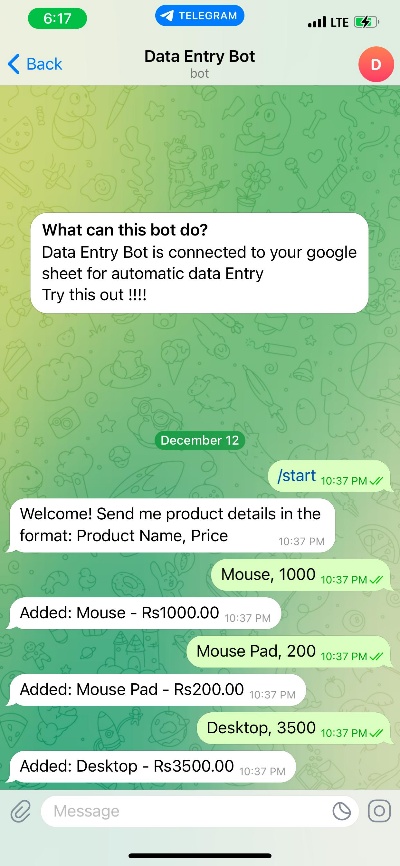
*await application.run\_polling()*

*# Run the bot*

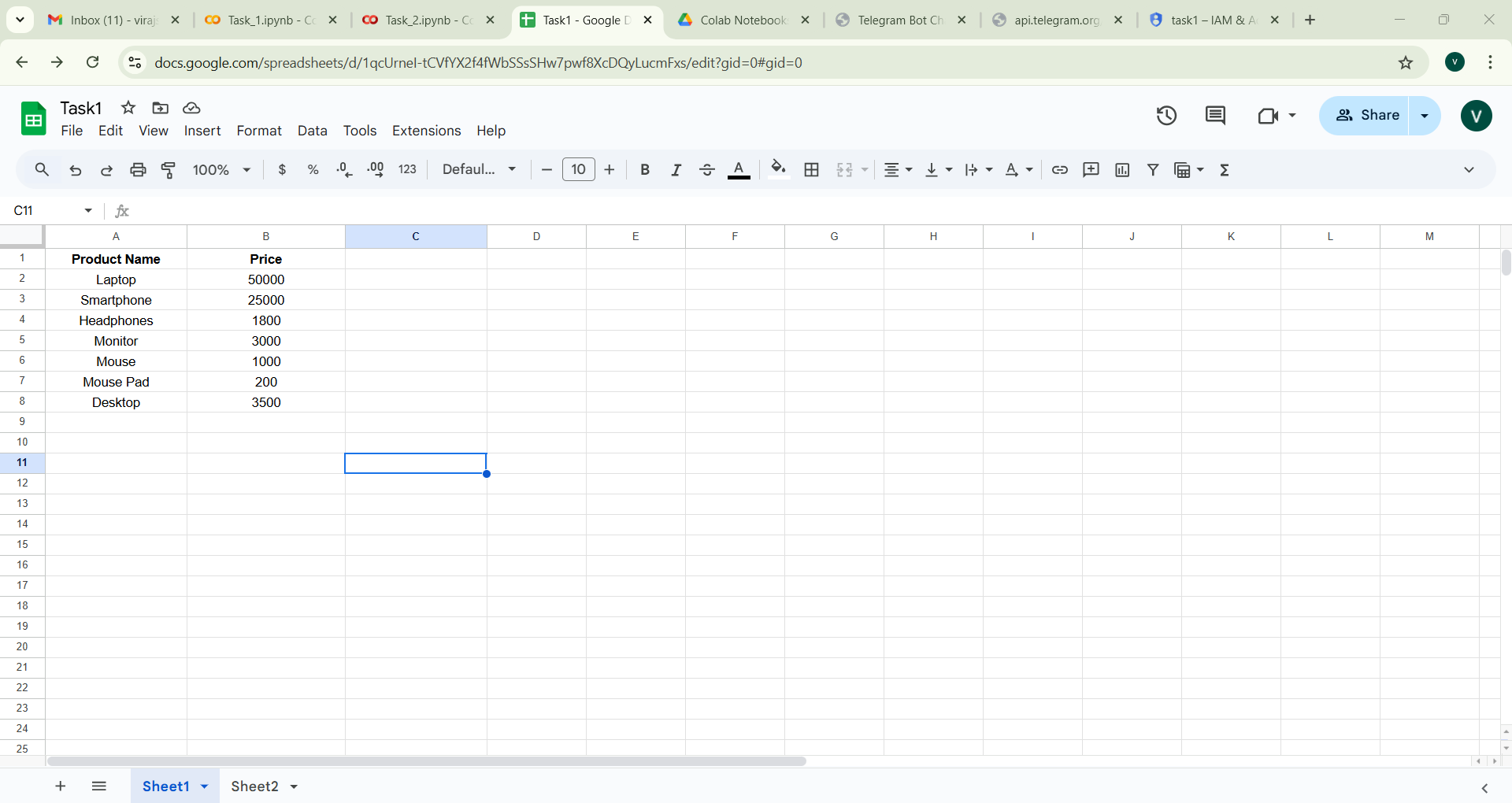
*await main()*

**Output :-**

Entered Data from telegram Bot.



Output in Google Sheet.



**Conclusion :-**

In conclusion, this task shows how simple tools like Telegram and Google Sheets can be combined to create powerful solutions. The ability to input product data effortlessly via a familiar platform like Telegram and have it seamlessly stored in Google Sheets feels like a small step toward making day-to-day tasks smarter and easier. Working on this project has been challenging , especially in understanding how APIs and asynchronous programming come together to deliver real-time functionality. It’s satisfying to see how a few lines of code can reduce manual effort, save time, and make data management accessible to anyone.