```
import os.path
import csv
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

# If modifying these scopes, delete the file token.json.
SCOPES = ["https://www.googleapis.com/auth/spreadsheets"]

# Dictionary mapping camera brands to their respective target
sheet numbers
camera_brand_sheets = {
    "Axis": "Axis",
    "Vivotek": "Vivotek",
    "Hanwha": "Hanwha-Samsung",
    "Pelco": "Pelco",
    "Bosch": "Bosch",
    "Avigilon": "Avigilon",
    "ICRealtime": "ICRealtime",
    "I-Pro TIGER": "I-Pro",
}

# The target spreadsheet ID
TARGET_SPREADSHEET_ID = "1a4TGw-
_HVxkisRxGGrDFo14Rszb-u5Qt9_RhMJ00Z8Q"


def authenticate_google_sheets():
    """Authenticate and return the Sheets API service object."""
    creds = None
    if os.path.exists("token.json"):
        creds =
Credentials.from_authorized_user_file("token.json", SCOPES)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                "client_secret_249463672757-
2ubs5u75uiinv8pf1ln15reke994t77m.apps.googleusercontent.c
```

```
om.json", SCOPES
            )
            creds = flow.run_local_server(port=0)
        with open("token.json", "w") as token:
            token.write(creds.to_json())

    return build("sheets", "v4", credentials=creds)


def read_camera_data(file_name):
    """Read and process camera data from a text file using CSV
parser."""
    data_to_write = []
    headers = {}

    try:
        with open(file_name, "r") as file:
            # Use CSV reader which handles quoted fields properly
            csv_reader = csv.reader(file)

            # Read header and determine available columns
            header = next(csv_reader)

            # Check for each type of information in the header
            headers = {
                'has_codec_info': 'Supported Codecs' in header and
'Current Codec' in header,
                'has_ptz_info': 'PTZ Capable' in header,
                'has_audio_info': 'Has Microphone' in header and 'Has
Audio Output' in header,
                'has_smart_search': 'Smart Search Support' in
header,
                'has_dewarp': 'Dewarp Capable' in header
            }

            # Read the rest of the lines
            for row in csv_reader:
                # Replace dashes or other placeholder text with
None
                row = [cell if cell != '-' else None for cell in row]
                data_to_write.append(row)

    except Exception as e:
```

```
        print(f"Error reading {file_name}: {e}")

    return data_to_write, headers


def write_to_google_sheet(service, data, sheet_name, headers):
    """Write data to the Google Sheet."""
    try:
        sheet = service.spreadsheets()

        # Get the existing header to check if we need to update it
        result =
sheet.values().get(spreadsheetId=TARGET_SPREADSHEET_ID,
range=f"'{sheet_name}'!1:1").execute()
        existing_headers = result.get('values', [[]])[0]

        # Build the complete header based on available data
        complete_header = ["IP ADDR", "Manufacturer", "Model",
"Firmware", "MAC/Serial Number"]

        # Add additional headers based on what's available in the
data file
        if headers.get('has_codec_info', False):
            complete_header.extend(["Supported Codecs", "Current
Codec"])

        if headers.get('has_ptz_info', False):
            complete_header.append("PTZ Capable")

        if headers.get('has_audio_info', False):
            complete_header.extend(["Has Microphone", "Has Audio
Output"])

        if headers.get('has_smart_search', False):
            complete_header.append("Smart Search Support")

        if headers.get('has_dewarp', False):
            complete_header.append("Dewarp Capable")

        # Add Notes at the end
        complete_header.append("Notes")

        # Check if we need to update the header
        header_needs_update = False
```

```
        for column in complete_header:
            if column not in existing_headers and column !=
"Notes":  # Notes might be at the end
                header_needs_update = True
                break

        # Update the header if needed
        if header_needs_update or not existing_headers:
            sheet.values().update(
                spreadsheetId=TARGET_SPREADSHEET_ID,
                range=f"'{sheet_name}'!A1",
                valueInputOption="RAW",
                body={"values": [complete_header]}
            ).execute()
            print(f"Header updated in {sheet_name}.")

        # Clear existing data to prevent stale entries
        sheet.values().clear(
            spreadsheetId=TARGET_SPREADSHEET_ID,
            range=f"'{sheet_name}'!A2:Z1000"  # Adjust range as
needed for your data volume
        ).execute()
        print(f"Cleared existing data in {sheet_name}.")

        # Write the camera data, starting from the second row
(row 2)
        if data:
            sheet.values().update(
                spreadsheetId=TARGET_SPREADSHEET_ID,
                range=f"'{sheet_name}'!A2",
                valueInputOption="RAW",
                body={"values": data},
            ).execute()
            print(f"Data successfully written to {sheet_name}.")
        else:
            print(f"No data to write to {sheet_name}.")

    except HttpError as err:
        print(f"An error occurred: {err}")


def main():
    print("Starting Enhanced Camera Data Upload to Google
```

```
Sheets v2.0")

    # Authenticate Google Sheets API
    service = authenticate_google_sheets()

    # Loop through each camera brand's file
    for brand, sheet_name in camera_brand_sheets.items():
        print(f"Processing data for {brand} cameras...")

        # Read the camera data from the text file
        file_name = f"{brand}CameraInformation.txt"  # Assume
the file name follows the format

        # Check if file exists before attempting to read
        if not os.path.exists(file_name):
            print(f"File {file_name} not found. Skipping {brand}.")
            continue

        camera_data, headers = read_camera_data(file_name)

        if camera_data:
            # Write the data to the appropriate sheet
            write_to_google_sheet(service, camera_data,
sheet_name, headers)
        else:
            print(f"No data found for {brand} in {file_name}.")


if __name__ == "__main__":
    main()
```