Generation of Final Validation Data

Import Libraries

In [221…

```
from azure.storage.blob import BlobServiceClient, BlobSasPermissions, generate_b
from msrest.authentication import ApiKeyCredentials
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionP
from datetime import datetime, timedelta, time, date
import unicodedata
import pandas as pd
from urllib.parse import quote
from time import sleep
import re
import io, json
from azure.ai.formrecognizer import DocumentAnalysisClient
from azure.core.credentials import AzureKeyCredential
```

Creation of required functions

In [222…

```
def sanitize(text: str) -> str:
    """
    Convierte texto a ASCII plano, minúsculas y
    reemplaza cualquier cosa que no sea [a-z0-9] por "_".
    """
    text = unicodedata.normalize("NFKD", text)         # quita tildes
    text = text.encode("ascii", "ignore").decode()
    text = text.lower()
    text = re.sub(r"[^a-z0-9]+", "_", text)            # → '_' por esp./punto/et
    text = re.sub(r"_{2,}", "_", text).strip("_")       # colapsa '__' y bordes
    return text
```

In [223…

```
def sanitize_for_compare(text: str) -> str:
    """
    Reutiliza sanitize() y además elimina los '_' generados
    para que Radha S. Kumar, Radha S Kumar y radha_s_kumar
    queden en la misma forma «radha s kumar».
    """
    # 1) Reutiliza tu sanitize de blobs
    txt = sanitize(text)        # ej. 'radha_s_kumar'
    # 2) Sustituye '_' por espacio y colapsa dobles espacios
    txt = txt.replace("_", " ")
    txt = " ".join(txt.split())
    return txt.strip()
```

In [224…

```
def make_sas_url(container: str, blob: str, hours: int = 2) -> str:
    """
    Devuelve una URL firmada (SAS) de lectura para el blob indicado.
    """
    sas_token = generate_blob_sas(
        account_name      = ACCOUNT_URL.split("https://")[1].split(".")[0],
        container_name    = container,
        blob_name         = blob,
        account_key       = ACCOUNT_KEY,
        permission        = BlobSasPermissions(read=True),
        expiry            = datetime.utcnow() + timedelta(hours=hours),
```

```
    )
    encoded_blob = quote(blob, safe="")          # espai → %20, «.» es deixa
    return f"{ACCOUNT_URL.rstrip('/')}/{container}/{encoded_blob}?{sas_token}"
```

In [225…

```
def extract_name_from_id(id_url: str, client) -> tuple[str, str]:
    """
    Devuelve (first_name, last_name) del documento de identidad
    usando el modelo prebuilt-idDocument.
    """
    # 1. Lanza la inferencia
    poller  = client.begin_analyze_document_from_url(
        model_id="prebuilt-idDocument",
        document_url=id_url,
        locale="en-US"
    )
    doc     = poller.result().documents[0]
    fields  = doc.fields

    # 2. Normaliza los alias de nombre / apellido
    first_aliases = ("FirstName", "FirstNames", "GivenName", "GivenNames")
    last_aliases  = ("LastName", "LastNames", "Surname", "FamilyName")

    first_name = next(
        (fields[a].value for a in first_aliases if a in fields and fields[a].val
        ""
    )
    last_name = next(
        (fields[a].value for a in last_aliases if a in fields and fields[a].valu
        ""
    )
    dob   = fields.get("DateOfBirth", None)
    dob   = dob.value if dob else None        # datetime.date ó None

    return first_name.strip().title(), last_name.strip().title(), dob
```

In [226…

```
def parse_boarding_pass(bp_url: str, client) -> dict:
    """
    Lanza el modelo custom 'Boarding_pass_data_extractor'
    y devuelve un diccionario con los campos clave.
    """
    poller = client.begin_analyze_document_from_url(
        model_id="Boarding_pass_data_extractor",
        document_url=bp_url,
        locale="en-US",
    )
    result = poller.result()

    if not result.documents:
        raise ValueError("No se detectó ningún documento")

    doc_fields = result.documents[0].fields

    # Saca los valores que te interesan
    passenger_name = doc_fields["PassengerName"].value                    # «
    carrier        = doc_fields["Carrier"].value                          # «
    flight_no      = doc_fields["Flight No."].value                       # «
    seat           = doc_fields["Seat"].value                             # «
    origin         = doc_fields["From"].value                             # «
    destination    = doc_fields["To"].value                               # «
```

```
    date_str       = doc_fields["Date"].value                                  # «
    boarding_time  = doc_fields["Boarding Time"].value                         # «
    flight_class   = doc_fields["Class"].value                               # «Eco

    # Divide el nombre si lo necesitas por separado
    # (ajusta si el formato es Lastname Firstname o viceversa)
    first_name, last_name = passenger_name.split(maxsplit=1)

    return {
        "first_name": first_name,
        "last_name":  last_name,
        "carrier":    carrier,
        "flight_no":  flight_no,
        "seat":       seat,
        "class":      flight_class,
        "origin":     origin,
        "destination":destination,
        "date":       date_str,
        "boarding_time": boarding_time,
    }
```

In [227…

```
def norm_text(text):
    import unicodedata
    if not isinstance(text, str):
        text = str(text)  # ← esto previene el error
    return unicodedata.normalize("NFKD", text).encode("ascii", "ignore").decode(
```

In [228…

```
def to_date(s: str) -> date:
    """
    Convierte una cadena a datetime.date.
    Soporta varios formatos como:
    - "2025-06-29"
    - "29 June 2025"
    - "April 20, 2022"
    - "Apr 20, 2022"
    - "20 Apr 2022"
    """
    formatos = [
        "%Y-%m-%d",     # 2025-06-29
        "%d %B %Y",     # 29 June 2025
        "%B %d, %Y",    # April 20, 2022
        "%b %d, %Y",    # Apr 20, 2022
        "%d %b %Y",     # 20 Apr 2022
    ]
    for fmt in formatos:
        try:
            return datetime.strptime(s.strip(), fmt).date()
        except ValueError:
            continue
    raise ValueError(f"Formato de fecha desconocido: {s}")
```

In [229…

```
def to_time(s: str) -> time:
    """
    Convierte una cadena a datetime.time.
    Ignora un posible sufijo de zona (“PST”, “CEST”…).
    Soporta:
      - 14:30
      - 4:30 PM
      - 10:00 AM PST   → 10:00 AM
```

```
    """
    if s is None:
        raise ValueError("Hora vacía")

    # 1) quita zona horaria al final (2-4 letras)
    clean = re.sub(r"\b[A-Z]{2,4}$", "", s.strip()).strip()

    # 2) prueba varios formatos
    for fmt in ("%H:%M", "%I:%M %p"):
        try:
            return datetime.strptime(clean, fmt).time()
        except ValueError:
            continue

    raise ValueError(f"Formato de hora desconocido: {s}")
```

Authentication

In [ ]:
```
ACCOUNT_URL  = "https://001finalproject.blob.core.windows.net/"
ACCOUNT_KEY  = "KEY"
CONTAINER_NAME = "step5"
BLOB_NAME      = "flightManifest.csv"

CONTAINER_STEP2 = "step2"

CONTAINER_STEP4 = "step4"
ENDPOINT      = 'https://pythonejercicio-prediction.cognitiveservices.azure.com/
PREDICTION_KEY = 'KEY'
PROJECT_ID     = "9b1fa97e-0db3-46ee-93b4-9eead57c48d2"
PUBLISHED_NAME = "Iteration2"

ID_PREFIX       = "digital_id_"
BP_PREFIX       = "boarding-"
lighter_prefix = "lighter-"

doc_endpoint = "https://document-intelligence-udacity.cognitiveservices.azure.co
doc_key      = "KEY"
```

First Step:

we access to blob storage to get flightmanifest.csv to know the users that I am going to compare and later on I will update the document

In [231…
```
blob_service = BlobServiceClient(account_url=ACCOUNT_URL,
                                 credential=ACCOUNT_KEY)
container = blob_service.get_container_client(CONTAINER_NAME)

df_manifest = None

for blob in container.list_blobs():

    if not blob.name.lower().endswith(".csv"):
        continue
    if BLOB_NAME.lower() not in blob.name.lower():
        continue

    blob_client = blob_service.get_blob_client(container=CONTAINER_NAME,
```

```
                                                    blob=blob.name)
    csv_bytes = blob_client.download_blob().readall()
    df_manifest = pd.read_csv(io.BytesIO(csv_bytes))
    print(f"✅ '{blob.name}' cargado ({len(df_manifest)} filas).")
    break

if df_manifest is None:
    raise FileNotFoundError(f"No se encontró {BLOB_NAME} en {CONTAINER_NAME}")

df_manifest.tail(10)
```

✅ 'FlightManifest.csv' cargado (7 filas).

Out[231…

	Passanger Name	First name	Last Name	Date of Birth	Sex	Carrier	Flight No.	Class	From	
0	Sameer Kumar	Sameer	Kumar	25 January 1990	M	UA	234	Economy	San Francisco	Ch
1	Radha S. Kumar	Radha	S. Kumar	05 March 1994	F	UA	234	Economy	San Francisco	Ch
2	James Webb	James	Webb	15 December 1970	M	UA	234	Business	San Francisco	Ch
3	Libby Herold	Libby	Herold	10 February 1996	F	UA	234	Business	San Francisco	Ch
4	James Jackson	James	Jackson	12 October 1956	M	UA	234	Economy	San Francisco	Ch
5	Avkash Chauhan	Avkash	Chauhan	01 January 1990	M	UA	234	Economy	San Francisco	Ch
6	Alberto León	Alberto	León	10 October 1986	M	UA	234	Economy	San Francisco	Ch

7 rows × 21 columns

In [232…

```
df_manifest.columns
```

Out[232…

```
Index(['Passanger Name', 'First name', 'Last Name', 'Date of Birth', 'Sex',
       'Carrier', 'Flight No.', 'Class', 'From', 'To', 'Date', 'Baggage',
       'Seat', 'Gate', 'Boarding Time', 'Ticket No.', 'NameValidation',
       'DoBValidation', 'PersonValidation', 'BoardingPassValidation',
       'LuggageValidation'],
      dtype='object')
```

3-Way Person Name Validation + DoB Validation + Boarding Pass Validation + Ligther validation:

For the lighter validation, I uploaded some foto without and with lighters, I added the name of the owner of the Tray to simulate this step

In [236…

```
doc_client = DocumentAnalysisClient(
    endpoint   = doc_endpoint,
    credential = AzureKeyCredential(doc_key)
)

credentials = ApiKeyCredentials(in_headers={"Prediction-key": PREDICTION_KEY})
predictor   = CustomVisionPredictionClient(ENDPOINT, credentials)

Threshold = 0.25

CLASS_MAP = {
    "E": "ECONOMY",
    "B": "BUSINESS",
    "F": "FIRST",
    "P": "PREMIUM ECONOMY",
    # puedes añadir más si las hay
}

for idx, row in df_manifest.iterrows():

    manifest_first = row["First name"].strip().title()
    manifest_last  = row["Last Name"].strip().title()
    manifest_DoB    = datetime.strptime(row["Date of Birth"].strip(), "%d %B %Y"

    # Generem els noms de fitxer dins de /step2
    fname = sanitize(manifest_first)
    lname = sanitize(manifest_last)

    id_blob = f"{ID_PREFIX}{fname}_{lname}.pdf"
    bp_blob = f"{BP_PREFIX}{fname}-{lname}.pdf"
    lighter_blob = f"{lighter_prefix}{fname}.jpg"

    id_url = make_sas_url(CONTAINER_STEP2, id_blob)  # SAS de lectura
    bp_url = make_sas_url(CONTAINER_STEP2, bp_blob)
    lighter_url= make_sas_url(CONTAINER_STEP4, lighter_blob)

    try:

        # --- 1. ID ---
        id_first, id_last, id_dob = extract_name_from_id(id_url, doc_client)

        # --- 2. Boarding Pass ---
        bp_data   = parse_boarding_pass(bp_url, doc_client)
        bp_first  = bp_data["first_name"].title()
        bp_last   = bp_data["last_name"].title()


        # --- 3. 3-Way comparison ---
        id_name = sanitize_for_compare(f"{id_first} {id_last}")
        bp_name = sanitize_for_compare(f"{bp_first} {bp_last}")
        mf_name = sanitize_for_compare(f"{manifest_first} {manifest_last}")

        if id_name == bp_name == mf_name:
            df_manifest.at[idx, "NameValidation"] = True
            print(f"✅  {manifest_first} {manifest_last} 3-Way Person Name Vali
        else:
            print("❌  Mismatch:")
            print("    • Manifest:", mf_name)
            print("    • ID      :", id_name)
```

```
        print("    • Boarding :", bp_name)

    # --- 4. Date of Birth Validation ---
    if id_dob and manifest_DoB:
        # 1) pasa SIEMPRE a string ISO ('YYYY-MM-DD')
        id_dob_str = (
            id_dob.isoformat() if isinstance(id_dob, date)   # date → '1990-
            else str(id_dob)                                 # ya era string
        )
        manifest_dob_str = manifest_DoB.isoformat()  # date → '2022-04-20'
        if id_dob_str == manifest_dob_str:
            df_manifest.at[idx, "DoBValidation"] = True
            print(f"✅  {manifest_first} {manifest_last} Date of Birth Vali
        else:
            print(f"❌  Date of Birth mismatch: ID={id_dob_str}, Manifest={

    # --- 5. Boarding Pass Validation ---
    # Boarding Pass data
    bp_flight   = bp_data['flight_no'].strip().upper()
    bp_seat     = bp_data["seat"].strip().upper()
    bp_class    = bp_data["class"].strip().upper()
    bp_class = CLASS_MAP.get(bp_class, bp_class)
    bp_origin   = norm_text(bp_data["origin"])
    bp_dest     = norm_text(bp_data["destination"])
    bp_date     = to_date(bp_data["date"])                # datetime.date
    bp_time     = to_time(bp_data["boarding_time"])
    #Boarding Pass data validation
    mf_flight   = norm_text(row["Flight No."]).replace(" ", "").upper()
    mf_seat     = row["Seat"].strip().upper()
    mf_class    = row["Class"].strip().upper()
    mf_origin   = norm_text(row["From"])
    mf_dest     = norm_text(row["To"])
    mf_date     = to_date(row["Date"])
    mf_time     = to_time(row["Boarding Time"])

    #Comparison

    flight_ok = bp_flight == mf_flight
    seat_ok   = bp_seat   == mf_seat
    class_ok  = bp_class  == mf_class
    orig_ok   = bp_origin == mf_origin
    dest_ok   = bp_dest   == mf_dest
    date_ok   = bp_date   == mf_date
    time_ok   = bp_time   == mf_time

    boarding_valid = all([flight_ok, seat_ok, class_ok,
                          orig_ok, dest_ok, date_ok, time_ok])

    # Update Manifest
    if boarding_valid:
        df_manifest.at[idx, "BoardingPassValidation"] = True
        print(f"✅  {manifest_first} {manifest_last} Boarding Pass Validate
    else:
        print("❌  Boarding-Pass mismatch:")
        if not flight_ok: print(f"  • Flight #      : {bp_flight} ≠ {mf_flig
        if not seat_ok:   print(f"  • Seat          : {bp_seat} ≠ {mf_seat}"
        if not class_ok:  print(f"  • Class         : {bp_class} ≠ {mf_class
        if not orig_ok:   print(f"  • Origin        : {bp_origin} ≠ {mf_orig
        if not dest_ok:   print(f"  • Destination   : {bp_dest} ≠ {mf_dest}"
        if not date_ok:   print(f"  • Flight Date   : {bp_date} ≠ {mf_date}"
```

```
        if not time_ok:   print(f"  • Boarding Time : {bp_time} ≠ {mf_time}"

    # --- 6. Identity Validation ---
    # At this point, we assume the identity has been validated, I will chang
    # later on I will do the validation of my ID and image extracted from th
    # to show how the validation is done

    df_manifest.at[idx, "PersonValidation"] = True
    print(f"✅  {manifest_first} {manifest_last} Identity Validated")

    # --- 7. Lighter Validation ---
    detection = predictor.detect_image_url(
            project_id     = PROJECT_ID,
            published_name = PUBLISHED_NAME,
            url            = lighter_url
        )

    hits = [
        p for p in detection.predictions
        if p.tag_name.lower() == "lighter" and p.probability >= Threshold
    ]

    if hits:

        print(f"🚫  {manifest_first} {manifest_last}: lighter FOUND ({hits[(
    else:
        df_manifest.at[idx, "LuggageValidation"] = True
        print(f"✅  {manifest_first} {manifest_last}: no lighter detected")

    # --- 8. Message on kiosk ---

    full_name = f"{manifest_first} {manifest_last}"

    flight_msg = (
        f"Dear Sir/Madam, {full_name},\n"
        f"You are welcome to flight # {mf_flight} leaving at {mf_time.strfti
        f"from {mf_origin} to {mf_dest}.\n"
        f"Your seat number is {mf_seat}, and it is confirmed."
    )

    # Verificamos cada validación
    name_ok     = df_manifest.at[idx, "NameValidation"]         == True
    dob_ok      = df_manifest.at[idx, "DoBValidation"]          == True
    boarding_ok = df_manifest.at[idx, "BoardingPassValidation"] == True
    luggage_ok  = df_manifest.at[idx, "LuggageValidation"]      == True

    # 1. ✅ Todo validado
    if name_ok and dob_ok and boarding_ok and luggage_ok:
        print("\n🟢 MENSAJE:")
        print(f"{flight_msg}\n"
            f"We did not find a prohibited item (lighter) in your carry-on b
            f"thanks for following the procedure.\n"
            f"Your identity is verified so please board the plane.")

    # 2. ❌ Lighter detectado
    elif name_ok and dob_ok and boarding_ok and not luggage_ok:
        print("\n🔴 MENSAJE:")
        print(f"{flight_msg}\n"
            f"We have found a prohibited item in your carry-on baggage, and
            f"Your identity is verified. However, your baggage verification
```

```
        # 3. ❌ ID fallida pero todo lo demás OK
        elif not name_ok and not dob_ok and boarding_ok and luggage_ok:
            print("\n🔴 MENSAJE:")
            print(f"{flight_msg}\n"
                f"We did not find a prohibited item (lighter) in your carry-on b
                f"Thanks for following the procedure.\n"
                f"Your identity could not be verified. Please see a customer ser

        # 4. ❌ Boarding pass incorrecto pero lo demás OK
        elif name_ok and dob_ok and not boarding_ok:
            print("\n🔴 MENSAJE:")
            print("Dear Sir/Madam,\n"
                "Some of the information in your boarding pass does not match th
                "so you cannot board the plane.\n"
                "Please see a customer service representative.")

        # 5. ❌ ID incorrecta pero boarding pass correcto
        elif boarding_ok and not (name_ok and dob_ok):
            print("\n🔴 MENSAJE:")
            print("Dear Sir/Madam,\n"
                "Some of the information on your ID card does not match the flig
                "so you cannot board the plane.\n"
                "Please see a customer service representative.")

        # 6. Otros casos mixtos
        else:
            print("\n🔴 MENSAJE:")
            print("Dear Sir/Madam,\n"
                "There was an issue validating your information. Please see a cu

    except Exception as e:
        print(f"⚠️  Error {manifest_first} {manifest_last}: {e}")
```

✅ Sameer Kumar 3-Way Person Name Validated
✅ Sameer Kumar Date of Birth Validated
✅ Sameer Kumar Boarding Pass Validated
✅ Sameer Kumar Identity Validated
✅ Sameer Kumar: no lighter detected

🟢 MENSAJE:
Dear Sir/Madam, Sameer Kumar,
You are welcome to flight # 234 leaving at 10:00 AM from san francisco to chicag
o.
Your seat number is 34A, and it is confirmed.
We did not find a prohibited item (lighter) in your carry-on baggage,
thanks for following the procedure.
Your identity is verified so please board the plane.
✅ Radha S. Kumar 3-Way Person Name Validated
✅ Radha S. Kumar Date of Birth Validated
✅ Radha S. Kumar Boarding Pass Validated
✅ Radha S. Kumar Identity Validated
🚫 Radha S. Kumar: lighter FOUND (63.4%)

🔴 MENSAJE:
Dear Sir/Madam, Radha S. Kumar,
You are welcome to flight # 234 leaving at 10:00 AM from san francisco to chicag
o.
Your seat number is 34B, and it is confirmed.
We have found a prohibited item in your carry-on baggage, and it is flagged for r
emoval.

Your identity is verified. However, your baggage verification failed, so please s
ee a customer service representative.
✅ James Webb 3-Way Person Name Validated
✅ James Webb Date of Birth Validated
✅ James Webb Boarding Pass Validated
✅ James Webb Identity Validated
✅ James Webb: no lighter detected

🟢 MENSAJE:
Dear Sir/Madam, James Webb,
You are welcome to flight # 234 leaving at 10:00 AM from san francisco to chicag
o.
Your seat number is 1A, and it is confirmed.
We did not find a prohibited item (lighter) in your carry-on baggage,
thanks for following the procedure.
Your identity is verified so please board the plane.
⚠️ Error Libby Herold: (403) Out of call volume quota for FormRecognizer F0 pri
cing tier. Please retry after 10 days. To increase your call volume switch to a p
aid tier.
Code: 403
Message: Out of call volume quota for FormRecognizer F0 pricing tier. Please retr
y after 10 days. To increase your call volume switch to a paid tier.
⚠️ Error James Jackson: (403) Out of call volume quota for FormRecognizer F0 pr
icing tier. Please retry after 10 days. To increase your call volume switch to a
paid tier.
Code: 403
Message: Out of call volume quota for FormRecognizer F0 pricing tier. Please retr
y after 10 days. To increase your call volume switch to a paid tier.
⚠️ Error Avkash Chauhan: (403) Out of call volume quota for FormRecognizer F0 p
ricing tier. Please retry after 10 days. To increase your call volume switch to a
paid tier.
Code: 403

```
Message: Out of call volume quota for FormRecognizer F0 pricing tier. Please retr
y after 10 days. To increase your call volume switch to a paid tier.
⚠️  Error Alberto León: (403) Out of call volume quota for FormRecognizer F0 pri
cing tier. Please retry after 10 days. To increase your call volume switch to a p
aid tier.
Code: 403
Message: Out of call volume quota for FormRecognizer F0 pricing tier. Please retr
y after 10 days. To increase your call volume switch to a paid tier.
```