Streamlit Invoice Validator - Complete Code Explanation

> **What This App Does:**
> Think of this as a quality checker for invoices before they go into your company's accounting system. It's like having a careful assistant who reviews invoices to catch mistakes before they cause problems.

Line-by-Line Explanation

1. Setting Up (Lines 1-3)

```
import streamlit as st
import pandas as pd
```

streamlit (st): This is like a toolbox that helps you build web apps without needing to be a web developer. We nickname it "st" for short typing.

pandas (pd): This is a tool for working with data tables (like Excel spreadsheets). We nickname it "pd".

2. Configuring the Page (Lines 4-6)

```
st.set_page_config(page_title="AP Invoice
```

```
Validator", page_icon="📄")
st.title("📄 AP Invoice Validation Dashboard")
st.write("Upload invoice data to validate before ERP
processing.")
```

set_page_config: Sets up your browser tab name and icon (that little 📄 you see)

title: Creates a big heading at the top of your page

write: Adds explanatory text (like instructions for users)

3. File Upload Section (Lines 8-12)

```
uploaded_file = st.file_uploader(
    "Upload Invoice CSV",
    type=["csv"]
)
```

This creates a button that lets users upload a CSV file (a spreadsheet file). Think of it like a "Choose File" button you've seen on websites. It only accepts CSV files, not Word docs or images.

4. Processing the Uploaded File (Lines 14-17)

```
if uploaded_file:
    df = pd.read_csv(uploaded_file)
    st.subheader("📊 Uploaded Data Preview")
    st.dataframe(df)
```

if uploaded_file: "If someone uploaded a file, do the

following..."

pd.read_csv: Opens the CSV file and turns it into a data table called "df"

st.dataframe(df): Shows the data table on screen so users can see what they uploaded

5. Defining Requirements (Lines 19-25)

```
required_columns = [
    "invoice_id",
    "supplier",
    "business_unit",
    "invoice_amount"
]
```

This is a checklist of columns (like Excel column headers) that **must** be in the file. It's like saying "Every invoice needs an ID, supplier name, business unit, and amount."

6. Checking for Missing Columns (Lines 27-34)

```
missing_columns = [
    col for col in required_columns if col not in
df.columns
]

if missing_columns:
    st.error(f"❌ Missing columns: {',
'.join(missing_columns)}")
```

This checks if any required columns are missing. If your file doesn't have "supplier" for example, it'll show a red error

message saying "❌ Missing columns: supplier"

7. Validation Logic (Lines 36-40)

```
else:
    df["error_reason"] = ""
    df.loc[df["supplier"].isna(), "error_reason"] +=
"Missing Supplier; "
    df.loc[df["business_unit"].isna(),
"error_reason"] += "Missing Business Unit; "
    df.loc[df["invoice_amount"] <= 0,
"error_reason"] += "Invalid Invoice Amount; "
```

If all columns are present, it creates a new column called "error_reason" and checks each row:

- Is the supplier name missing? → Add note: "Missing Supplier"
- Is the business unit missing? → Add note: "Missing Business Unit"
- Is the invoice amount zero or negative? → Add note: "Invalid Invoice Amount"

8. Separating Good from Bad Records (Lines 42-44)

```
valid_df = df[df["error_reason"] == ""]
error_df = df[df["error_reason"] != ""]
```

valid_df: All invoices with NO errors (error_reason is empty)

error_df: All invoices WITH errors (error_reason has text)

9. Displaying Results Side-by-Side (Lines 46-54)

```
col1, col2 = st.columns(2)
with col1:
    st.success(f"✅ Valid Records: {len(valid_df)}")
    st.dataframe(valid_df)
with col2:
    st.error(f"❌ Error Records: {len(error_df)}")
    st.dataframe(error_df)
```

Creates two columns (like splitting the screen in half):

- **Left side:** Shows valid invoices in green with a checkmark ✅
- **Right side:** Shows problem invoices in red with an X ❌

10. Download Button (Lines 56-62)

```
if not error_df.empty:
    st.download_button(
        "⬇️ Download Error Report",
        error_df.to_csv(index=False),
        file_name="invoice_errors.csv",
        mime="text/csv"
    )
```

If there ARE errors, creates a download button so users can save the error list as a CSV file to fix later.

Real-World Example

Imagine you upload this file:

invoice_id	supplier	business_unit	invoice_amount
001	ABC Corp	Sales	5000
002	*(empty)*	Marketing	3000
003	XYZ Inc	IT	-100

The app would show:

- ✅ **1 Valid:** Invoice 001
- ❌ **2 Errors:**
  - Invoice 002: "Missing Supplier"
  - Invoice 003: "Invalid Invoice Amount"

You could then download the error report to fix invoices 002 and 003!

Key Benefits

- **Prevents Errors:** Catches mistakes before they enter your accounting system
- **Saves Time:** Automatically checks hundreds of invoices in seconds
- **Easy to Use:** Just upload a file and see results instantly
- **Exportable:** Download error reports to share with your team