CSV to JSON Web Converter – Full Technical Documentation

1. Introduction

In today's data-driven world, converting between different data formats is a common need. While data often comes in **CSV (Comma-Separated Values)** format due to its simplicity, modern APIs and systems prefer the structured and hierarchical nature of **JSON (JavaScript Object Notation)**.

This project demonstrates a complete, deployable application for converting CSV files into JSON format using a combination of:

- **Flask** – a lightweight Python web framework (backend)
- **Streamlit** – a modern interface for building web apps with Python (frontend)
- **Pandas** – a robust library for reading and processing CSV data

---

2. Requirements

The required libraries are listed in `requirements.txt`:

Library	Purpose
`flask==3.1.0`	Web API creation
`flask-cors==5.0.1`	Manages CORS to allow frontend-backend communication
`streamlit==1.44.0`	Builds interactive frontend dashboards with Python
`requests==2.32.3`	Makes HTTP requests from frontend to backend
`pandas==2.2.3`	Handles reading CSVs and converting data to JSON format

Install with:

```
pip install -r requirements.txt
```

---

3. Backend – `flask_app.py`

This script defines the API endpoint that receives a CSV file and returns its JSON representation.

3.1 Imports

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import pandas as pd
import io
```

- `Flask`: Starts the web server
- `CORS`: Allows requests from different origins (e.g., frontend)
- `pandas`: Handles CSV reading and transformation
- `io`: Reads files into memory without saving them

3.2 Initialize App

```
app = Flask(__name__)
CORS(app)
```

Enables CORS and creates a Flask instance.

3.3 Define Endpoint: `/convert`

```
@app.route('/convert', methods=['POST'])
def convert_csv_to_json():
```

This route receives POST requests with a CSV file attached.

3.4 Handle Uploaded File

```
if 'file' not in request.files:
    return jsonify({"error": "No file uploaded"}), 400
file = request.files['file']
if file.filename == '':
    return jsonify({"error": "Invalid file"}), 400
```

Checks if the file was uploaded and has a valid name.

3.5 Convert CSV to JSON

```
csv_data = io.StringIO(file.stream.read().decode('utf-8-sig'))
df = pd.read_csv(csv_data)
json_list = df.to_dict(orient='records')
return jsonify({"json": json_list})
```

- Decodes the stream with `utf-8-sig` to avoid BOM issues

- Converts to DataFrame, then to a JSON-friendly list of dictionaries

---

3.6 Error Handling

```
except pd.errors.EmptyDataError:
    return jsonify({"error": "CSV file is empty"}), 400
except Exception as e:
    return jsonify({"error": str(e)}), 500
```

Handles empty files and other unexpected errors.

3.7 Start the Server

```
if __name__ == '__main__':
    import os
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)
```

Runs on all IP addresses, with a default port of `5000`.

---

4. Frontend – `streamlit_app.py`

A user interface built with Streamlit that allows CSV file upload and displays the converted JSON.

4.1 Imports

```
import streamlit as st
import requests
import json
```

Used for interface, HTTP communication, and JSON handling.

4.2 UI Title

```
st.title("CSV to JSON Converter")
```

Sets the main page title.

4.3 Project Info

```
with st.expander("📘 Project Details"):
    st.markdown("""...""")
```

Shows application documentation in a collapsible block.

4.4 Upload CSV

```
uploaded_file = st.file_uploader("Upload CSV file", type="csv")
```

Allows users to upload CSV files from their device.

4.5 Send File to Backend

```
response = requests.post(
    'https://csv-to-json-converter-backend.onrender.com/convert',
    files={'file': (uploaded_file.name, uploaded_file, 'text/csv')}
)
```

Sends the file to the Flask API for conversion.

4.6 Display Result

```
json_data = response.json().get('json')
st.json(json_data)
```

Shows the returned JSON in a pretty-printed format.

4.7 Download JSON

```
json_str = json.dumps(json_data, indent=2)
st.download_button(
    label="Save JSON",
    data=json_str,
    file_name=uploaded_file.name.replace(".csv", ".json"),
    mime="application/json"
)
```

Offers a download button for the resulting JSON file.

4.8 Error Handling

Handles both API errors and connection issues:

```
except requests.exceptions.ConnectionError:
    st.error("API server is not running.")
except Exception as e:
    st.error(f"Unexpected error: {e}")
```

---

5. Workflow Summary

```
CSV Upload (User)
      ↓
Streamlit sends file to Flask API
      ↓
Flask reads, processes, and returns JSON
      ↓
Streamlit displays and enables JSON download
```

---

6. Development Suggestions

- **Authentication** (JWT, API keys)
- **Schema validation** for structured CSV
- **Data preview** before conversion
- **Docker containers** for deployment
- **CSV cleaning tools** (missing data handling)

---

7. Conclusion

This project illustrates how a simple architecture combining **Flask** and **Streamlit** can create a powerful yet accessible tool for converting CSV data to JSON. It bridges the gap between raw data formats and structured, API-ready outputs without requiring any command-line knowledge from the user.

Key Takeaways:

- Easy CSV upload and conversion via web interface
- Reliable and clean data transformation using `pandas`
- Suitable for developers, analysts, and casual users
- Can be extended for enterprise-grade pipelines