

**Step 1: User Uploads JSON Data**

1. Upload a **JSON file** containing sales or structured data.
2. JSON is **preprocessed** (preprocessing.py).
3. Extracts **columns and sample values**.
4. Ensures **column names** are formatted correctly to avoid schema mismatches.

**Step 2: User Enters a Query**

1. Enter a natural language query like:

* *"Show sales trends by product category*

1. Also have chart type selection (optional).

**Step 3: OpenAI Generates Chart Schema**

**Call to LLM (openai\_api.py):**

* LLM receives **query, sample data, column names, and chart type**.
* Uses a **strict system prompt** (prompts.py) to enforce schema rules:
  + **Vega-Lite JSON Schema:** <https://vega.github.io/schema/vega-lite/v5.json>
  + **Plotly JSON Reference:** <https://plotly.com/javascript/reference/>
  + **ECharts JSON Docs:** <https://echarts.apache.org/en/option.html>

**Step 4: Schema Validation**

**In validation.py**:

* Checks if **Vega-Lite, Plotly, or ECharts** schema is **valid**.
* Uses jsonschema for **Vega-Lite** validation.
* Custom validation functions for **Plotly** and **ECharts**

**If valid → Proceed to Next step**

**If invalid → Send correction request to LLM** (up to **2 retries**).

**Step 5: Display Chart in Streamlit**

**If schema is valid:**

* **Displays chart** using Altair (for Vega-Lite), streamlit\_echarts (for E-charts) and plotly.graph\_object (for Plotly charts)
* Offers **download option** (JSON).

**If schema is invalid after retries:**

* Displays **error message** to the user.

**Step 6: Download Final Chart Schema**

* Users can **download the corrected Vega-Lite/Plotly/ECharts JSON schema**.