Converting Figma JSON to Offline Vector Database with FAISS

(figma to json convertion was done using postman get request)

using **Sentence-Transformers** and **FAISS**.

---

1. Objective

- Extract readable text (like labels or titles) from d Figma design.
- Convert those texts into **vectors** (numerical form).
- Store the vectors in a **vector database** (FAISS) for **semantic search**.

---

2. Technology

Tool/Lib	Purpose
`json` (built-in)	Parse the Figma design JSON file.
`SentenceTransformers`	Convert text into vector embeddings (offline, pre-trained model).
`FAISS`	Facebook AI Similarity Search. Used to store and search vectors fast.
`pickle`	Stores text metadata (ID + text) linked to vectors.
`Python`	Orchestrates the whole pipeline.

---

3. Folder Structure

```
figma-vector-db/
├── data/
│   ├── figma_data.json          # Input JSON from Figma
│   ├── text_nodes.json          # Extracted text nodes
│   ├── embedded_nodes.json      # Embeddings + metadata
│   ├── faiss.index              # FAISS index file
│   └── faiss_meta.pkl           # Metadata (IDs + text)
├── scripts/
│   ├── extract_nodes.py         # Step 1: JSON traversal
│   ├── embed_data.py            # Step 2: Generate embeddings
│   └── build_faiss_index.py     # Step 3: Build and query FAISS
├── requirements.txt             # Dependencies
├── venv/                        # Virtual environment
```

4. Step

Step 1: `extract_nodes.py`

- Traverses the Figma JSON tree (recursively).
- Collects all nodes with type `SHAPE_WITH_TEXT`.
- Extracts fields: `id`, `name`, `characters`
- Writes to `text_nodes.json`.

Important logic:

```
if node.get("type") == "SHAPE_WITH_TEXT":
    text = node.get("characters", "").strip()
```

Step 2: `embed_data.py`

- Loads `text_nodes.json`
- Uses `SentenceTransformer('all-MiniLM-L6-v2')`
- Converts each text into a **384-dimensional vector**.
- Fully offline (no LLM, no internet).
- Saves list of dictionaries:

```
[
  {
    "id": "1:2",
    "name": "Title",
    "text": "Dashboard",
    "embedding": [0.021, -0.189, ..., 0.132]
  }
]
```

Important model:

```
SentenceTransformer('all-MiniLM-L6-v2')
```

- Light, fast, good accuracy.

Step 3: `build_faiss_index.py`

- Loads `embedded_nodes.json`
- Extracts:
- Vectors → NumPy array (`float32`)
- IDs → for identification
- Text → for search display
- Builds FAISS index:

```
index = faiss.IndexFlatL2(dim)  # dim = 384
index.add(vectors)
```

- Stores metadata using `pickle`:

```
pickle.dump({"ids": [...], "texts": [...]}, f)
```

---

5. Search Example

When you run:

```
python scripts/build_faiss_index.py query "Dashboard"
```

- The query is embedded like your text.
- FAISS finds nearest vectors based on **Euclidean (L2) distance**.

Output:

```
Search results for: “Dashboard”
• 1:2 [0.0000] → Dashboard
• 1:10 [0.9516] → Analytics Section
• 1:106 [1.2363] → User Profile
```

@@@@Lower distance = more similar@@@@@

- Shows the closest text chunks by meaning

---

6. use case

You can now:

- Search your Figma design by concept (not exact text)
- Build AI-powered design tools
- Detect duplicated labels or inconsistent UIs
- Build UI auto-suggestions or design summarizers

---

7. Optional Improvement can be made

Feature	Tools / How
Add web UI	Use Streamlit or FastAPI

Feature	Tools / How
Fuzzy text search	Combine FAISS with keyword search
Update index dynamically	Watch folders / re-index with timestamp checks
Group similar components	Use clustering (e.g. KMeans on embeddings)

---

8. Note

- Ensure all inputs are **UTF-8 encoded**
- Sentence-Transformer embeddings are **not normalized** → use `IndexFlatL2` or normalize manually
- Embeddings are **not 100% accurate** → always validate results visually

---