

Feature: Semantic Duplicate & Conflict Detector

Goal

Detect:

1.  Exact duplicates
2.  Near duplicates (minor edits)
3.  Conflicting drafts (same topic, different content)

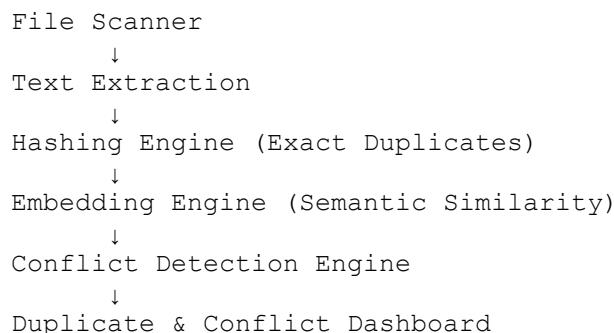
Example:

- Resume_Final.docx
- Resume_NewSkills.docx

System says:

“Both files describe your resume but contain different skill sets. Conflict detected.”

Full Architecture



Complete Tech Stack

Since you’re already building other features:

Backend

- FastAPI
- Python

◆ Database

- SQLite

◆ Hashing

- hashlib (built-in)

◆ Embeddings

- sentence-transformers
- Model: all-MiniLM-L6-v2

◆ Similarity Search

- FAISS

◆ Conflict Detection (Advanced)

- NLI model (optional)
- Or rule-based contradiction detection

For hackathon → embeddings + rule-based conflict detection is enough.

🔍 STEP 1: Exact Duplicate Detection (Hashing)

Use SHA256.

```
import hashlib

def generate_hash(file_path):
    with open(file_path, "rb") as f:
        file_bytes = f.read()
    return hashlib.sha256(file_bytes).hexdigest()
```

Store hash in DB.

If two files have same hash → exact duplicate.

That's easy.

STEP 2: Near Duplicate Detection (Semantic Similarity)

This is where AI comes in.

Step 2.1 Extract Text

Use:

File	Library
PDF	PyMuPDF
DOCX	python-docx
PPTX	python-pptx
TXT	open()

Combine:

- filename
 - extracted text
 - folder name
-

Step 2.2 Convert to Embeddings

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')
embedding = model.encode(file_text)
```

Each file becomes vector.

Step 2.3 Compare Similarity

Cosine similarity:

```
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity([embedding1], [embedding2])
```

If:

`similarity > 0.85`

→ Near duplicate.

Δ STEP 3: Conflict Detection

This is more advanced.

Near duplicate means:

“Almost same”

Conflict means:

“Same topic, different meaning”

Approach 1 (Simple & Hackathon-Friendly)

If similarity between:

`0.65 < similarity < 0.85`

And filename contains:

- final
- updated
- v2
- draft

→ Flag as “Potential Conflict”

Approach 2 (Better — Using Sentence-Level Comparison)

Break document into sentences.

Compare sentence embeddings.

If many sentences differ significantly:

→ Conflict detected.

Approach 3 (Advanced NLI Model — Optional)

Use Natural Language Inference model:

Check if one document contradicts another.

But this may be heavy for hackathon.

So rule-based + semantic difference is enough.



Conflict Scoring Strategy

```
if similarity > 0.9:  
    exact_duplicate  
elif similarity > 0.8:  
    near_duplicate  
elif similarity > 0.6 and topic_match:  
    conflict
```



How to Detect “Topic Match”

Simple trick:

Extract keywords using TF-IDF.

If both contain:

- resume
- skills
- experience

→ same topic.



STEP 4: Store Relationships

Create table:

```
CREATE TABLE file_relationships (  
    file1_id INTEGER,  
    file2_id INTEGER,  
    relationship_type TEXT,  
    similarity_score REAL  
) ;
```

Types:

- exact_duplicate
 - near_duplicate
 - conflict
-

Dashboard UI

Show:

Duplicates Found

- 3 exact duplicates
- 5 near duplicates

Conflicts Detected

- Resume_V1 vs Resume_Final
- Project_Report_Draft vs Project_Report_Final

Add action buttons:

- Delete duplicate
 - Compare versions
 - Merge
-

Bonus Feature (Judges Love This)

Add visual diff.

For text files:

Show side-by-side comparison.

Highlight:

- Added skills
- Removed experience

That makes it look powerful.



Folder Structure

```
duplicate-detector/
  ├── backend/
  │   ├── hashing_engine.py
  │   ├── embedding_engine.py
  │   ├── similarity_engine.py
  │   ├── conflict_detector.py
  │   └── database.py
  └── frontend/
      ├── DuplicateDashboard.jsx
      └── ConflictViewer.jsx
```



Execution Plan

Phase 1 – Must Have

- Exact duplicate detection (hash)
 - Semantic similarity > threshold
 - Show duplicate list
-

Phase 2 – Conflict Detection

- Mid-range similarity detection
 - Filename rule boost
 - Topic similarity detection
-

Phase 3 – WOW Layer

- Side-by-side comparison
 - Merge suggestion
 - Auto-delete suggestions
-



Why This Is Hackathon Strong

Because:

Most systems detect only exact duplicates.

You detect:

- Meaning duplicates
- Version conflicts
- Draft inconsistencies

That's next-level.



Combine With Other Features

Now your full system becomes:

Smart Search
Why Do I Have This
Hot/Warm/Cold Engine
Compression Optimizer
Backup Buddy
Duplicate & Conflict Detector

This is no longer a feature.

This is:

AI-Powered Personal Storage Operating System



Killer Presentation Line

Say:

“We don’t just detect identical files. We detect identical ideas.”

That line hits HARD.



Practical Advice For Hackathon

You don’t need:

- Perfect conflict detection

- Large-scale vector DB

Use:

- 20–30 sample files
- Show powerful demo scenario

Focus on storytelling.



Final Recommended Stack For YOU

FastAPI
SQLite
SentenceTransformers
FAISS
hashlib
React

Keep everything modular.