🧠 Feature: Hot vs Cold Memory Brain Engine

🎯 Goal

Classify files like human memory:

Memory Type	Meaning	Action
🔥 Hot	Frequently used	Keep on fast storage
🌥 Warm	Occasionally used	Normal storage
❄ Cold	Rarely used	Compress / Archive

🏗 System Architecture

```
File Scanner
    ↓
Usage Tracker
    ↓
Feature Extraction
    ↓
Memory Classification Engine
    ↓
Recommendation Engine
    ↓
Heatmap Visualization
```

🛠 Complete Tech Stack

Since you’re already building Smart Search & Why-Do-I-Have-This:

🔷 Backend

- **FastAPI**
- Python

🔷 Database

- SQLite

🔷 Visualization

- React + Chart.js
  or
- Python + Plotly (if simple demo)

🔷 Optional

- Scikit-learn (if clustering)

For hackathon → Rule-based classification is enough.

🧠 Core Idea

Memory temperature depends on:

1. Last accessed time
2. Access frequency
3. Recency trend
4. File size (optional)

🔍 Step 1: Track Usage Data

You already need this for previous features.

Store:

Field	Meaning
file_id	Unique ID
open_count	Number of times opened
last_accessed	Timestamp
created_at	Timestamp
size	File size

Database schema:

```
CREATE TABLE files (
    id INTEGER PRIMARY KEY,
    name TEXT,
    path TEXT,
    created_at DATETIME,
    last_accessed DATETIME,
    open_count INTEGER,
    size INTEGER
);
```

🔥 Step 2: Calculate “Heat Score”

We create a memory score.

Formula Approach (Simple & Effective)

```
recency_score = 1 / (days_since_last_access + 1)
frequency_score = log(open_count + 1)
size_weight = log(size + 1)

heat_score =
    0.5 * recency_score
  + 0.4 * frequency_score
  + 0.1 * size_weight
```

Normalize scores between 0 and 1.

🧠 Step 3: Classify Into Hot / Warm / Cold

Rule-Based Classification

```
if heat_score > 0.7:
    memory_type = "Hot"
elif heat_score > 0.3:
    memory_type = "Warm"
else:
    memory_type = "Cold"
```

That’s enough for hackathon.

🧠 Step 4: Optional ML Clustering (Advanced)

Instead of rules, you can cluster:

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)
kmeans.fit(feature_vectors)
```

Clusters automatically form:

- High usage
- Medium
- Low

Then label them as:

- Hot
- Warm
- Cold

Judges will appreciate this.

🧠 Step 5: Recommendation Engine

Now you add intelligence.

If Cold + Large File:

→ Recommend compression

If Cold + Old (> 1 year):

→ Suggest archive

If Hot:

→ Suggest SSD optimization

Example output:

“12 cold files detected. Compressing them can save 2.4GB.”

That aligns perfectly with SanDisk.

📊 Step 6: Heatmap Visualization

This is the WOW factor.

Option 1: Folder Heatmap

Show folders colored by temperature.

Folder	Color
Projects	🔥 Red
Downloads	❄ Blue

Option 2: Timeline Memory Map

Show usage across months:

```
Jan 🔥🔥🔥
Feb 🔥🔥
Mar 🌥
Apr ❄
```

Use:

- Chart.js
- Plotly

💻 Example Backend Logic

```
from datetime import datetime
import math

def calculate_heat(file):
    days = (datetime.now() - file["last_accessed"]).days

    recency_score = 1 / (days + 1)
    frequency_score = math.log(file["open_count"] + 1)
    size_score = math.log(file["size"] + 1)

    heat = 0.5*recency_score + 0.4*frequency_score + 0.1*size_score

    return heat
```

📦 Folder Structure

```
memory-brain/
│
├── backend/
│   ├── main.py
│   ├── heat_engine.py
│   ├── scanner.py
│   ├── tracker.py
│   ├── recommendation.py
│
├── frontend/
```

```
│   ├── Dashboard.jsx
│   ├── Heatmap.jsx
│
└── database/
```

🚀 Complete Implementation Plan

Phase 1 – Basic Version (Must Have)

- Track open count
- Track last accessed
- Calculate heat score
- Classify into 3 categories

Phase 2 – Visual Dashboard

- Display count of Hot/Warm/Cold
- Show pie chart
- Show heatmap by folder

Phase 3 – Intelligent Recommendations

- Detect cold large files
- Suggest compression
- Estimate space saving

🧠 Why This Wins Hackathons

Because it's:

- Emotionally relatable (human memory model)
- Technically strong (scoring + clustering)
- Storage optimization focused (SanDisk relevance)

🏆 Combine With Other Features

This feature becomes VERY powerful when combined with:

- Smart Search
- Why Do I Have This?
- Duplicate Finder

Then your system becomes:

AI Personal Memory Assistant

⚠️ Important Practical Advice

Since this is hackathon:

You don’t need real OS-level monitoring.

Instead:

- Simulate open_count
- Simulate access history
- Use sample dataset

Focus on demo quality.

🎯 What Judges Care About

Not perfect ML.

They care about:

- Clear idea
- Clean architecture
- Real use-case
- Good UI
- Logical scoring system

💡 Final Recommended Stack For YOU

Since you already:

- Know ML basics
- Have worked on backend APIs

Use:

```
FastAPI
SQLite
Scikit-learn (optional)
React + Chart.js
```

Keep it simple but polished.

🔥 Killer Demo Line

Say this in presentation:

“Just like the human brain forgets rarely used memories, our system intelligently identifies cold files and recommends compression — saving space automatically.”

That line alone creates impact.
