Workload Distribution

👤 Member 1 — Data Storage & Sampling Engineer

Responsibilities

1. **Data File Handling**
    - Understand and manage the binary file storing all vectors.
    - Ensure correct use of memory-mapped files.
    - Implement any helper functions needed for efficient row access.
2. **Sampling Module**
    - Write code to randomly sample vectors from the dataset.
    - Ensure sampling does not exceed RAM limits.
    - Prepare a sampled dataset for k-means training.
3. **Support for Index Building**
    - Build helper functions that allow the index builder (Member 3) to read vectors efficiently.

Your Output / Deliverables

- Efficient reading of individual vectors from disk.
- A working sampling function.
- Documentation explaining the data format and I/O strategy (memory-mapped approach).

👤 Member 2 — Clustering & Centroid Builder

Responsibilities

1. **Train k-Means (MiniBatchKMeans)**
    - Run mini-batch k-means on the sampled vectors.
    - Experiment with different numbers of clusters.
    - Choose the final cluster count (n_clusters).
2. **Centroid Management**
    - Write centroids to disk in a compact format.
    - Provide a function for loading centroids during retrieval.
3. **Index Directory Setup**
    - Define the structure where index files (centroids + inverted lists) will be stored.

Your Output / Deliverables

- A fully trained set of centroids saved to disk.
- Python code for training and loading centroids.
- Documentation explaining clustering strategy, sample size choice, and cluster count.

👤 Member 3 — Index Construction & Inverted Lists Engineer

Responsibilities

1. **Implement _build_index()**
    - Read every vector using functions from Member 1.
    - Find the nearest centroid for each vector (using centroids from Member 2).
    - Assign vector ID to the correct centroid.
2. **Create Inverted Lists**
    - One file per cluster.
    - Only store vector IDs to keep index size small.
    - Organize files efficiently for fast loading in retrieval.
3. **Index Size Optimization**
    - Make sure all inverted list files meet index size constraints.
    - Compress or optimize formats if needed.

Your Output / Deliverables

- Correct implementation of _build_index().
- All inverted list files generated and saved to disk.
- Documentation describing index creation and file-organization decisions.

👤 Member 4 — Retrieval Engineer (retrieve() + Evaluation)

Responsibilities

1. **Implement retrieve()**
    - Normalize query vector.
    - Load centroids from disk (allowed in RAM).
    - Compute distances to all centroids.
    - Select top nprobe clusters (e.g., top 3–10).
    - Load inverted list files for selected clusters.
    - Load only the needed vectors from the main file.
    - Compute cosine similarity for candidates.
    - Return top-K similar vectors.
2. **Performance Optimization**
    - Make sure retrieval meets RAM and time limits.
    - Profile slow steps and improve them.
3. **Evaluation**
    - Run evaluation notebook.
    - Test accuracy (recall) and timing.
    - Share results with the team to adjust parameters.

Your Output / Deliverables

- A fast, correct retrieve() function.
- Passing all constraints in the evaluation notebook.
- Documentation explaining the retrieval algorithm and optimization choices.

📌 Ultra-Clear Summary Table

Member	Main Role	What They Produce
1	Data Storage & Sampling	Efficient data access + sampling code
2	Clustering & Centroids	Trained centroids + centroid files on disk
3	Index Builder	Inverted lists + full _build_index()
4	Retrieval & Evaluation	retrieve() implementation + performance testing