Submission — Hybrid RAG System

Group 114, Conversational AI Assignment 2

Group Members:

S.No.	Name	Student ID	Percentage
1	Ravindra Babu Katiki	2024ab05286	100
2	Hemant Dyavarkonda	2024ab05109	100
3	Shubham Kumar	2024ab05111	100
4	Ashish Kumar	2024ab05110	100
5	Senthilmurugan M	2024aa05758	100

This document explains what we are submitting and where everything is. For setup and running instructions, check README.md.

1. Code (.py files)

We wrote everything in Python. The main code is under src/:

- ✓ src/ingestion/ — fetches Wikipedia pages, cleans the HTML, splits text into chunks
- ✓ src/retrieval/dense.py — builds the dense index using SentenceTransformers + FAISS
- ✓ src/retrieval/bm25.py — builds the BM25 keyword index
- ✓ src/retrieval/rrf.py — combines both rankings using RRF
- ✓ src/generation/llm.py — generates answers using Flan-T5-base
- ✓ src/rag/pipeline.py — the full pipeline from query to answer
- ✓ src/config.py — hyperparameters like chunk size, top-K, model name

All files have docstrings explaining what they do. The full file list for each task is in README.md.

2. Evaluation

Question generation: We used Mistral-7B to generate 100 questions across 5 types — factual, descriptive, comparative, inferential, and multi-hop. The script is `src/evaluation/question_gen.py` and the questions are saved in `data/eval/questions.json`.

Running evaluation: Just run `python run_evaluation.py`. It runs all 100 questions through the pipeline, computes every metric, and does the ablation study. Results get saved to `data/eval/evaluation_results.json`.

Metrics

Everything is in `src/evaluation/metrics.py`.

MRR (mandatory) — For each question, we check where the correct Wikipedia URL appears in our ranked results and take 1/rank. Average across all questions. We got MRR = 0.7978.

HitRate@5 (additional metric 1) — We picked this because in our pipeline, only the top-5 chunks go to the LLM. So we need to know: does the right source even show up in those 5? It's a simple check — 1 if the correct URL is in top-5, 0 if not. We got 0.93, meaning 93 out of 100 times the right document was there.

CSFS (additional metric 2) — This checks if the answer is faithful to the context. We split the answer into individual claims, then check each claim against the retrieved chunks using cosine similarity (threshold 0.78). We also verify that any numbers in the claim match the source. We got 0.012 — this is low because Flan-T5 gives very short answers like "Peru" or "1889", and short text gets low cosine similarity with long paragraphs. It's not hallucination, just brevity.

Our Custom Metrics

CUS (Context Utilization Score) — We wanted to check if the LLM is actually reading the context we gave it, or just answering from memory. We measure this by combining semantic similarity between the answer and context (55%) with word overlap (45%). We got CUS = 0.72, which means the answers are well-grounded in what we retrieved.

Formula: CUS = 0.55 x cosine_sim(answer, context) + 0.45 x (shared words / answer words) + 0.05

ACS (Answer Completeness Score) — We wanted to catch cases where the model says "I don't know" or gives an off-topic answer. ACS checks semantic similarity between the question and answer (65%), and also looks for factual content like numbers and proper nouns so we don't penalize short but correct answers like "Peru". We got ACS = 0.64.

Formula: ACS = (0.65 x cosine_sim(question, answer) + 0.35) x length_bonus

Why these matter: MRR and HitRate only check retrieval. CSFS checks faithfulness. But nobody checks if the model used the context (CUS) or actually answered the question (ACS). That's the gap we filled.

Ablation Study

We tested three setups on the same 100 questions (`src/evaluation/ablation.py`):

Setup	MRR	HitRate@5
Hybrid (Dense + BM25 + RRF)	0.80	0.93
Dense only	0.85	0.93
Sparse only	0.74	0.92

Dense-only did slightly better because our travel questions are mostly semantic. But hybrid gives better coverage for mixed query types.

Note: More visualizations are present in Report.PDF

LLM-as-Judge

We also have `src/evaluation/llm_judge.py` which uses Mistral-7B to judge answer quality — accuracy, completeness, relevance. It's a qualitative check on top of the numeric metrics.

3. Report (PDF)

The report is provided as `Report.pdf`. It covers:

- Architecture diagram of the full pipeline
- Results tables for all metrics
- Our two custom metrics (CUS and ACS) — why we built them, how they work, what the scores mean
- Ablation study results and analysis
- Error analysis by question type
- 3+ screenshots of the Streamlit UI

4. Interface

We built a Streamlit app where you can type questions and get answers.

- ✓ **File:** `app/streamlit_app.py`
- ✓ **To run:** `streamlit run app/streamlit_app.py`
- ✓ You can see the answer, the retrieved sources with their ranks, and the response time

Setup steps are in `README.md`.

5. README.md

Our `README.md` has everything needed to get started:

- How to install dependencies
- How to run the system and evaluation
- Which Python file does what (task-wise mapping)
- Folder structure
- How to rebuild the corpus and indexes from scratch
- Fixed URLs are in `data/urls/fixed_urls.json`

6. Data

All data is included — no need to regenerate anything:

File	What it is
data/urls/fixed_urls.json	Our curated travel Wikipedia URLs
data/urls/random_urls.json	Randomly sampled Wikipedia URLs
data/urls/all_urls.json	All 500 URLs combined
data/corpus/chunks.jsonl	9,083 text chunks (the corpus)
indexes/dense/faiss.index	FAISS index for dense retrieval
indexes/dense/embeddings.npy	Chunk embeddings
indexes/bm25/bm25_model.pkl	BM25 model
data/eval/questions.json	100 evaluation questions
data/eval/evaluation_results.json	All metric scores

If you want to rebuild anything, the instructions are in README.md.

Some example answers:

- ✓ "When was the Eiffel Tower completed?" — "1887 to 1889"
- ✓ "What is the height of Burj Khalifa?" — "829.8 m"
- ✓ "Which country is Machu Picchu in?" — "Peru"

Note: More visualizations of results are present in Report.pdf