🔍 Hybrid RAG System with Automated Evaluation# Hybrid RAG System with Automated Evaluation# Hybrid RAG System with Automated Evaluation# Hybrid RAG System with Automated Evaluation

GitHub Repository

Python 3.10+

Streamlit Dashboard GitHub Repository

> A comprehensive **Hybrid RAG System** combining Dense Vector Retrieval, Sparse Keyword Retrieval, and Reciprocal Rank Fusion for Wikipedia-based Question Answering.

📦 **Repository:** github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_EvaluationA comprehensive implementation of a **Hybrid RAG System** combining **Dense Vector Retrieval (ChromaDB)**, **Sparse Keyword Retrieval (BM25)**, and **Reciprocal Rank Fusion (RRF)** to answer questions from Wikipedia articles. GitHub Repository GitHub Repository

---

📑 Table of Contents**GitHub Repository:** https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

---

🚀 Quick Start**GitHub Repository:** https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation**GitHub Repository:** https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

🚀 Quick Start

Prerequisites

Prerequisites

| Requirement | Version |

|-------------|---------|- Python 3.10+

| Python | 3.10+ |

| RAM | 4GB+ |- 4GB+ RAM**Last Updated:** February 8, 2026---

| Disk Space | 2GB+ |

Installation

Installation

```
# 1. Clone repository

git clone
https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalua
tion.git

cd Hybrid_RAG_System_with_Automated_Evaluation```bash---## 🚀 Quick Start



# 2. Create virtual environment# Clone repository

python -m venv venv

source venv/bin/activate  # Windows: venv\Scripts\activategit clone
https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalua
tion.git



# 3. Install dependenciescd Hybrid_RAG_System_with_Automated_Evaluation

pip install -r requirements.txt

```## 🚀 Quick Start### Prerequisites
```
```

```
### Run Application# Create virtual environment


```bashpython -m venv venv- Python 3.10+

# Start Streamlit Dashboard

streamlit run app_chromadb.py --server.port 8502source venv/bin/activate
# Windows: venv\Scripts\activate
```

Prerequisites- 4GB+ RAM

🌐 **Access URL:** http://localhost:8502

Install dependencies

Run Evaluation

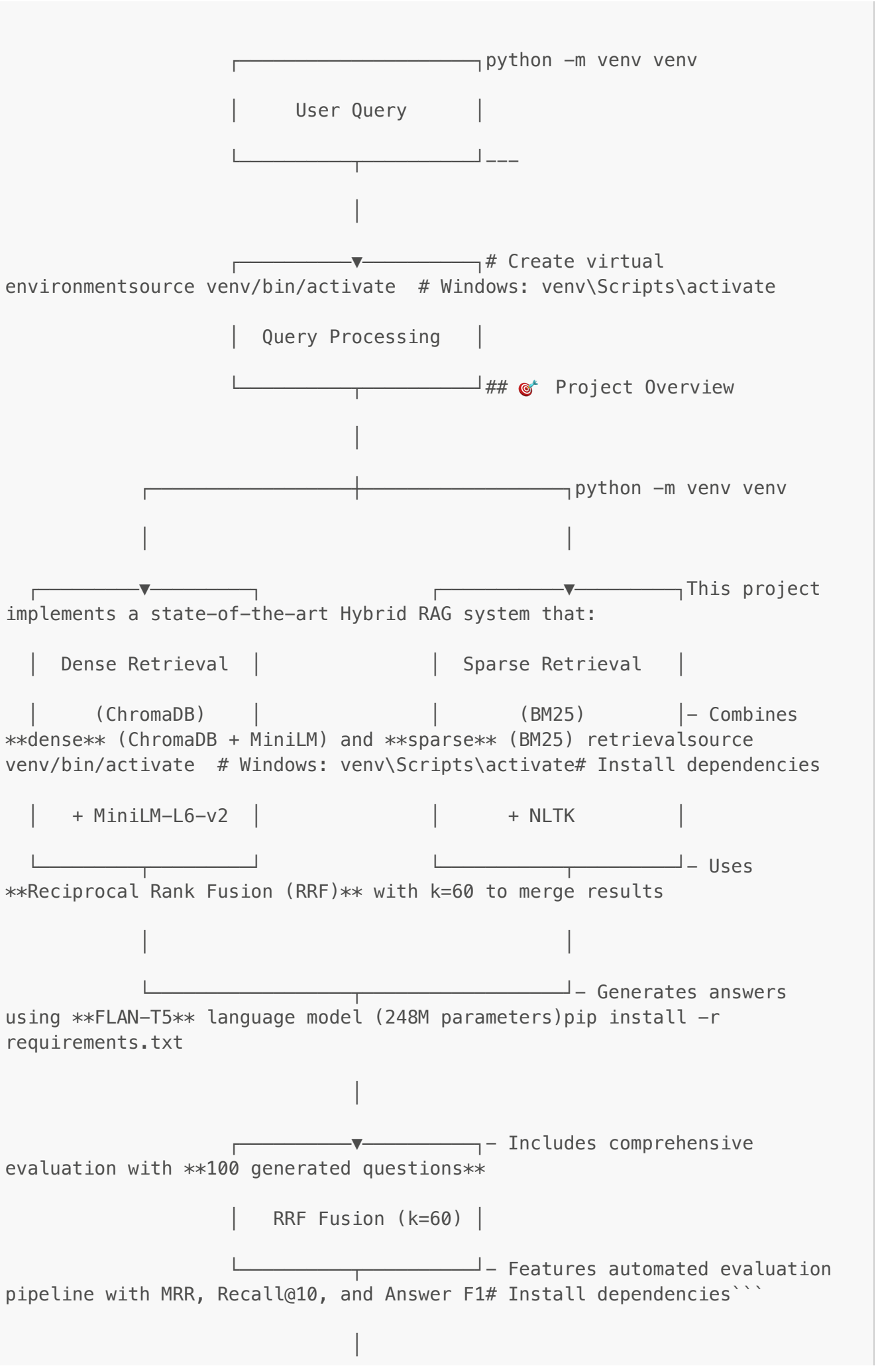
pip install -r requirements.txt- Python 3.10+

```
# Run full evaluation (100 questions × 3 methods)```

python evaluate_chromadb_fast.py

```- 4GB+ RAM### Installation


---### Run the Application

🎯 Project Overview

This project implements a **state-of-the-art Hybrid RAG system**
that:```bash

| Feature | Description |# Start Streamlit UI### Installation```bash

|---------|-------------|
```

```
| 🔷 **Dense Retrieval** | ChromaDB + MiniLM embeddings (384 dimensions)
|streamlit run app_chromadb.py --server.port 8502

| 🔶 **Sparse Retrieval** | BM25 with NLTK tokenization |

| 🔀 **Fusion** | Reciprocal Rank Fusion (RRF) with k=60 |```# Clone
repository

| 🤖 **Generation** | FLAN-T5-Base (248M parameters) |

| 📊 **Evaluation** | MRR, Recall@10, Answer F1 |

| 🖥️ **Interface** | Interactive Streamlit Dashboard |

**Access URL:** http://localhost:8502```bashgit clone
https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalua
tion.git

### 📈 Key Statistics



| Metric | Value |

|--------|-------|### Run Evaluation# Clone repositorycd
Hybrid_RAG_System_with_Automated_Evaluation

| Wikipedia URLs | 500 (200 fixed + 300 random) |

| Total Chunks | 7,519 segments |

| Embedding Dimensions | 384 |

| Chunk Size | 500 characters |```bashgit clone
https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalua
tion.git

| Evaluation Questions | 100 |

# Full evaluation (100 questions × 3 methods)

---

python evaluate_chromadb_fast.pycd
Hybrid_RAG_System_with_Automated_Evaluation# Create virtual environment

## 📊 System Architecture
```


python -m venv venv
User Query
Create virtual
environmentsource venv/bin/activate # Windows: venv\Scripts\activate
Query Processing
🎯 Project Overview
python -m venv venv
This project
implements a state-of-the-art Hybrid RAG system that:
Dense Retrieval
(ChromaDB)
+ MiniLM-L6-v2
Sparse Retrieval
(BM25)
+ NLTK
- Combines
dense (ChromaDB + MiniLM) and **sparse** (BM25) retrievalsource
venv/bin/activate # Windows: venv\Scripts\activate# Install dependencies
- Uses
Reciprocal Rank Fusion (RRF) with k=60 to merge results
- Generates answers
using **FLAN-T5** language model (248M parameters)pip install -r
requirements.txt
- Includes comprehensive
evaluation with **100 generated questions**
RRF Fusion (k=60)
- Features automated evaluation
pipeline with MRR, Recall@10, and Answer F1# Install dependencies```

```
                    ┌──────────▼──────────┐- Interactive dashboard with
**Dense/Sparse/Hybrid chunk comparison**
                    │   Top-K Chunks      │
                    └──────────┬──────────┘pip install -r requirements.txt
                               │
                    ┌──────────▼──────────┐### Key Statistics
                    │  Answer Generation  │
                    │   (FLAN-T5-Base)    │```### Run the Application
                    └──────────┬──────────┘
                               ||| Metric | Value |
                    ┌──────────▼──────────┐
                    │  Generated Answer   ||--------|-------|
                    │   + Source URLs     │
                    └─────────────────────┘| Total URLs | 500 (200 fixed +
300 random) |
```

| Total Chunks | 7,519 segments |### Run the Application```bash

---

| Embedding Dimensions | 384 |

📁 Project Structure

| Chunk Size | 500 characters |# Start Streamlit UI

```
Hybrid_RAG_System_with_Automated_Evaluation/| Evaluation Questions | 100 |
│
├── 📄 Core Files```bash./start_ui.sh
│   ├── chromadb_rag_system.py    # Core RAG implementation
│   ├── app_chromadb.py           # Streamlit UI---
│   ├── evaluate_chromadb_fast.py # Evaluation pipeline
```

```
│   ├── error_analysis.py         # Error analysis# Start Streamlit UI

│   └── api_chromadb.py           # REST API

│## 📊 System Architecture

├── 📁 data/

│   ├── fixed_urls.json           # 200 Wikipedia URLsstreamlit run
app_chromadb.py --server.port 8502# Or manually:

│   ├── corpus.json               # 7,519 chunks

│   └── questions_100.json        # 100 Q&A pairs```

│

├── 📁 evaluation/
┌──────────────────────────────────────────────────────────────┐```streamlit
run app_chromadb.py

│   ├── metrics.py                # MRR, Recall, F1

│   ├── comprehensive_metrics.py  # Extended metrics│
User Query                          │

│   ├── novel_metrics.py          # Custom metrics

│   ├── pipeline.py               # Eval
pipeline└──────────────────────────┬──────────────────────────────┘```

│   └── run_evaluation.py         # Runner script

│                           │

├── 📁 src/

│   ├── data_collection.py        # Wikipedia scraper
┌──────────┴──────────┐**Access URL:** http://localhost:8502

│   ├── semantic_chunker.py       # Chunking logic

│   ├── rrf_fusion.py             # RRF implementation              │
│

│   └── question_generation.py    # Q&A generation

│     ┌─────────▼─────────┐     ┌─────────▼─────────┐### Run Evaluation

├── 📁 docs/

│   ├── METRIC_JUSTIFICATION.md   # Metric rationale     │  Dense
Retrieval │     │ Sparse Retrieval │
```

```
│   ├── architecture_diagram.png  # System diagram
│   └── data_flow_diagram.png     # Data flow     │    (ChromaDB +   │
│     (BM25 +      │### Run Evaluation

│

├── 📁 screenshots/               # UI screenshots     │   MiniLM-L6-v2)
│     │      NLTK)       │

├── 📁 submission/                # Submission package

├── 📁 chroma_db/                 # Vector database
└──────────┬──────────┘     └──────────────────┬──────────┘```bash

│

├── 📄 requirements.txt           # Dependencies               │
│

├── 📄 config.yaml                # Configuration

└── 📄 README.md                  # This file
└──────────────┬──────────────┘```bash# Full evaluation (100 questions × 3
methods)
```

```
                    │
```

```
          ┌──────────────▼──────────────┐# Full evaluation (100 questions × 3
methods)python evaluate_chromadb_fast.py
```

📈 Evaluation Results

```
          │   Reciprocal Rank       │
```

🏆 Performance Summary

```
          │   Fusion (k=60)         │python evaluate_chromadb_fast.py
```

| Method | MRR | Recall@10 | Avg Time |

|--------|-----|----------|----------| └──────────────────────────┘

| Dense (ChromaDB) | 0.3025 | 0.33 | 5.86s |

| **Sparse (BM25)** | **0.4392** | **0.47** | **5.53s** | |```# Generate reports

| Hybrid (RRF) | 0.3783 | 0.43 | 6.37s |

```
                ┌──────────────▼──────────────┐
```

> 💡 **Key Finding:** BM25 outperforms Dense by **45%** on MRR for Wikipedia content.

```
                │     Top-K Chunks        │python generate_report.py
```

📊 Generation Metrics

```
                └──────────────┬──────────────┘
```

| Method | BLEU | ROUGE-L | BERTScore |

|--------|------|---------|----------| |---```

| Dense | 0.015 | 0.120 | 0.780 |

| Sparse | 0.022 | 0.145 | 0.820 | ┌──────────────▼──────────────┐

| Hybrid | 0.018 | 0.135 | 0.810 |

```
                │   Answer Generation     │
```

📋 Question Distribution

```
                │    (FLAN-T5-base)       │
```

| Type | Count | Percentage |

|------|-------|------------| └──────────────────────────┘## 🎯 Project Overview---

| Factual | 59 | 59% |

| Comparative | 15 | 15% | |

| Inferential | 11 | 11% |

| Multi-hop | 15 | 15% | ┌──────────────────▼──────────────────┐

| **Total** | **100** | 100% |

```
          |    Generated Answer     |
```

```
          |    + Source URLs        |This project implements a state-of-
the-art Hybrid RAG system that:## 🎯 Project Overview
```

📚 Documentation

```
          └─────────────────────────┘
```

| Document | Description | Link |

|----------|-------------|------|```- Combines **dense** (ChromaDB + MiniLM) and **sparse** (BM25) retrieval

| 📖 Submission Guide | Complete submission reference | SUBMISSION_REFERENCE_GUIDE.md |

| 📋 Deliverables | Assignment mapping | SUBMISSION_DELIVERABLES.md |

| 🔗 Quick Links | All GitHub links | QUICK_ACCESS_LINKS.md |

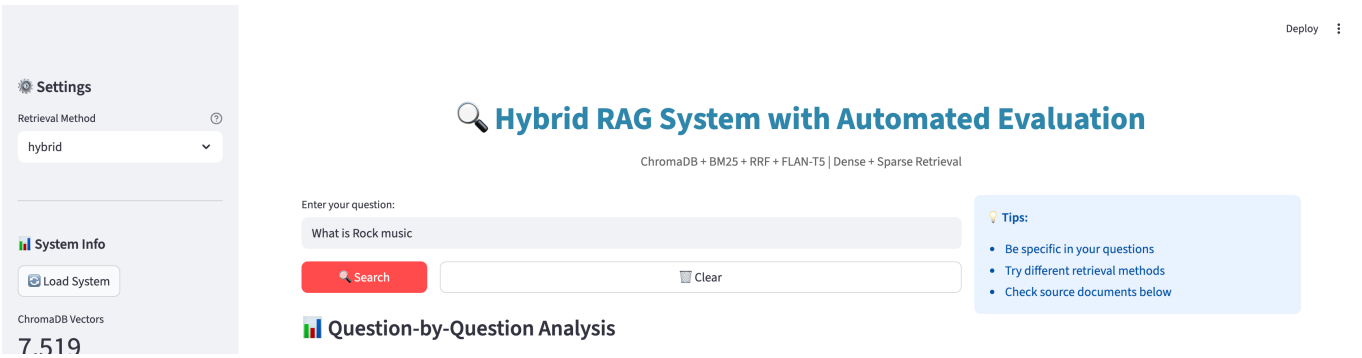
| 📊 Metrics | Metric justification | docs/METRIC_JUSTIFICATION.md |---- Uses **Reciprocal Rank Fusion (RRF)** with k=60 to merge resultsThis project implements a state-of-the-art Hybrid RAG system that:

| 📄 Report | Full evaluation report | submission/05_reports/Hybrid_RAG_Evaluation_Report.md |

🗂️ Project Structure- Generates answers using **FLAN-T5** language model (248M parameters)- Combines **dense** (ChromaDB + MiniLM) and **sparse** (BM25) retrieval

📸 Screenshots

Query Interface



BM25 Documents

7,519

Total Chunks

7,519

Recent Queries

1. What is Rock music...
2. What is Stock market...
3. What is Stock market...

Q1: What is Rock music

Answer: Rock music is a genre of popular music that originated in the United States as rock and roll in the ...

Retrieval Time: 0.075s | Generation Time: 3.911s

Top Score: 0.0328

Q2: What is Stock market

Answer: A stock market, equity market, or share market is the aggregation of buyers and sellers of stocks (a...

Retrieval Time: 0.078s | Generation Time: 5.288s

Top Score: 0.0328

Q3: What is Stock market

Answer: A stock market, equity market, or share market is the aggregation of buyers and sellers of stocks (a...

Retrieval Time: 0.072s | Generation Time: 5.691s

Top Score: 0.0328

Q4: What is AI

Answer: Artificial intelligence (AI) is the capability of computational systems to perform tasks typically a...

Retrieval Time: 0.071s | Generation Time: 3.948s

Top Score: 0.0164

Q5: What is AI

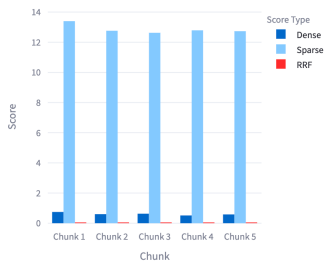
Answer: Artificial intelligence (AI) is the capability of computational systems to perform tasks typically a...

Retrieval Time: 0.070s | Generation Time: 4.412s

Top Score: 0.0164

📈 Chunk Score Visualization

Retrieval Scores by Chunk (Top 10)



🔍 Dense vs Sparse vs Hybrid Chunks

Dense Top 5 | Sparse Top 5 | Hybrid Top 5

Top 5 chunks by Hybrid (RRF Fusion)

▾ #1 - Score: 0.0328

Rock music

Source

Rock music is a genre of popular music that originated in the United States as rock and roll in the late 1940s and early 1950s, developing into a range of styles from the mid-1960s, primarily in the...

Dense: 0.7464 | Sparse: 13.3963 | RRF: 0.0328

▾ #2 - Score: 0.0315

Rock music

Source

In 1981, music journalist Robert Christgau said, the best rock jolts folk-art virtues directness, utility, natural audience into the present with shots of modern technology . Unlike many earlier styl...

Dense: 0.5998 | Sparse: 12.7603 | RRF: 0.0315

▸ #3 - Score: 0.0313

▸ #4 - Score: 0.0311

▸ #5 - Score: 0.0310

⚡ Performance Metrics

Method	Retrieval Time	Generation Time	Total Time
Hybrid	0.079s	3.510s	3.590s

📊 Retrieval Scores

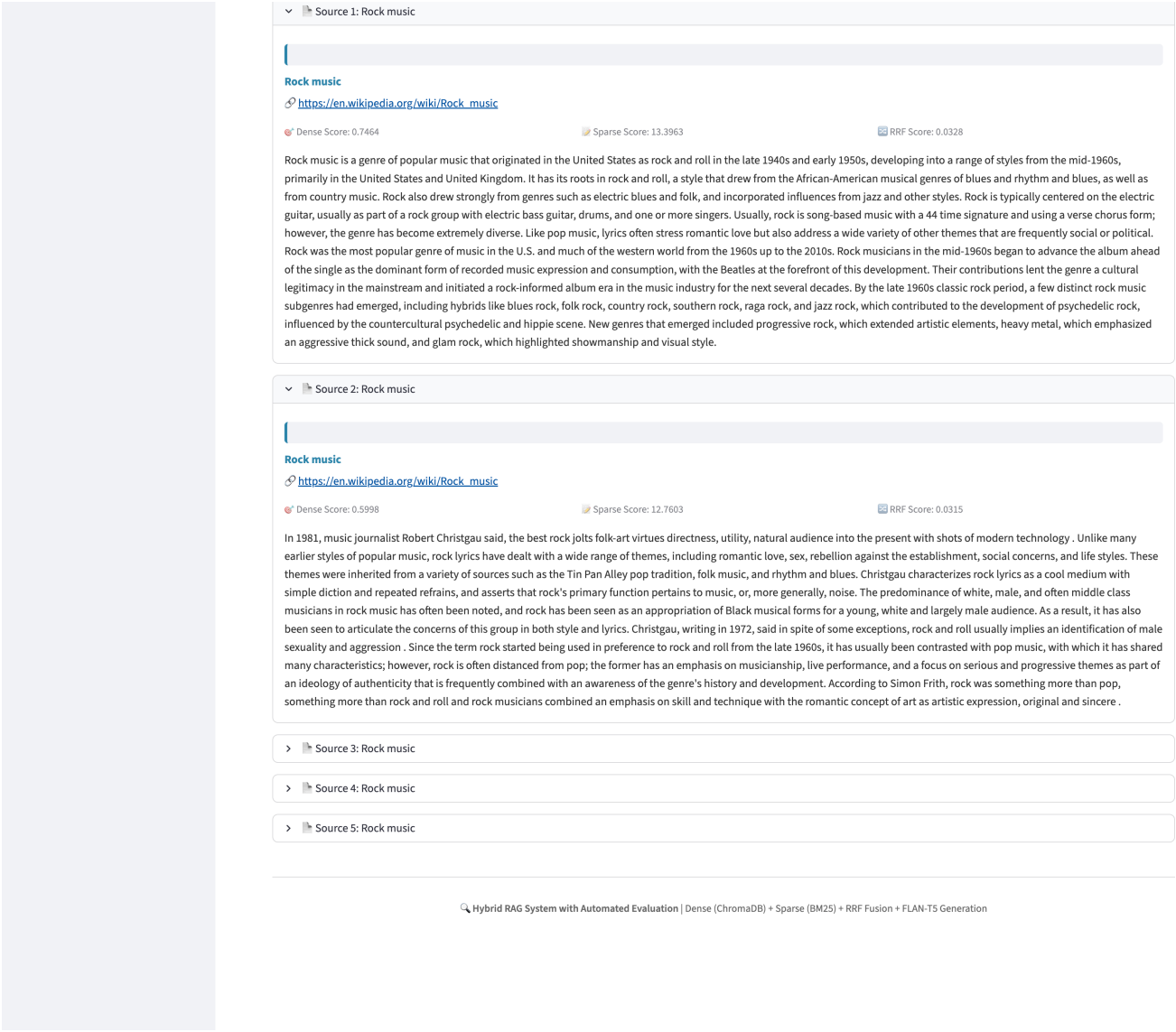
Avg Dense Score	Avg Sparse Score	Avg Hybrid Score	Top Score	Score Std Dev
0.6146	12.8605	0.0315	0.0328	0.0007
Vector similarity	BM25 score	RRF fusion	Best match	5 docs

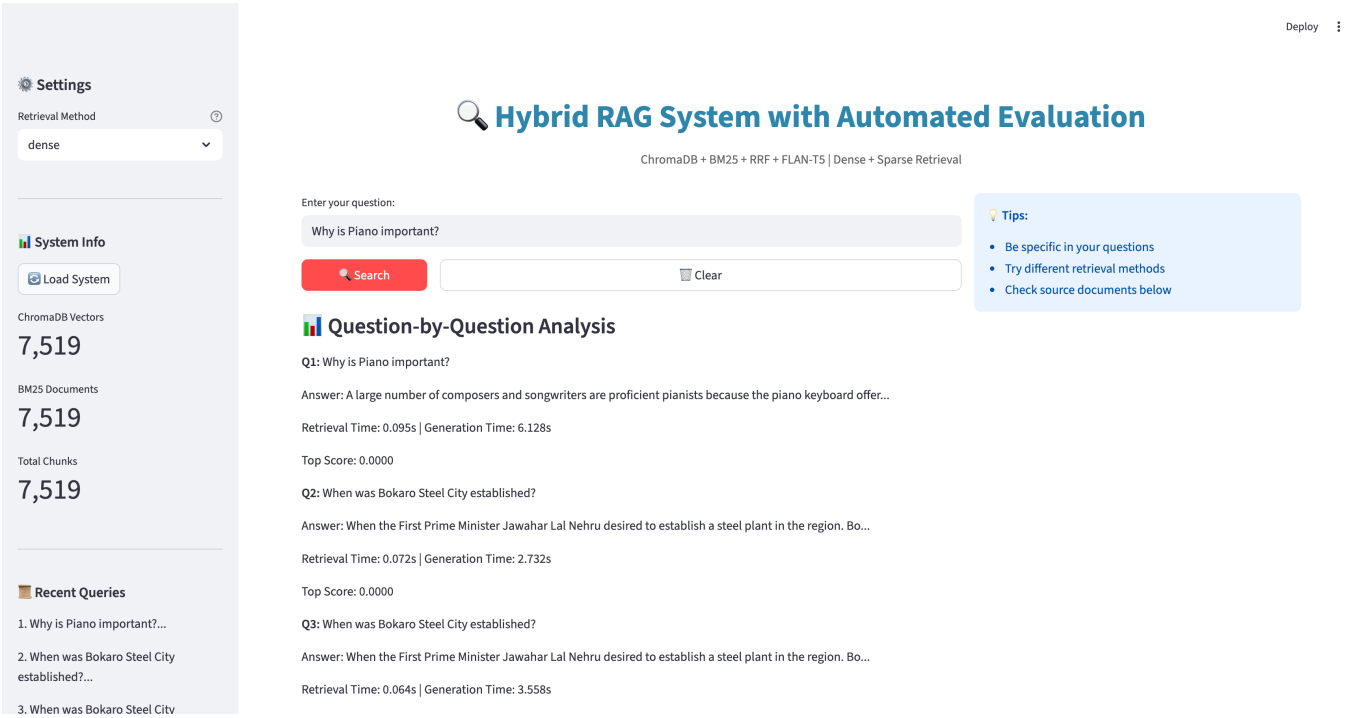
📝 Answer

> **Rock music is a genre of popular music that originated in the United States as rock and roll in the late 1940s and early 1950s, developing into a range of styles from the mid-1960s, primarily in the U.S. and United Kingdom.**

📚 Source Documents

☐ Show all sources




```- Includes comprehensive evaluation with **100 generated questions**- Uses **Reciprocal Rank Fusion (RRF)** with k=60 to merge results

Dense Retrieval ModeHybrid_RAG_System_with_Automated_Evaluation/


```

Top Score: 0.0000

Q4: Why is Piano important?

Answer: A large number of composers and songwriters are proficient pianists because the piano keyboard offer...

Retrieval Time: 0.080s | Generation Time: 4.861s

Top Score: 0.0000

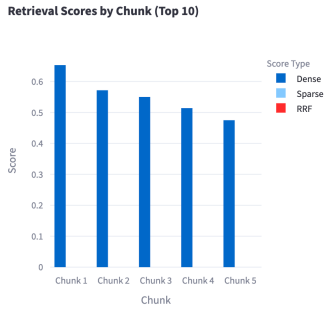
Q5: What is Rock music

Answer: Rock music is a genre of popular music that originated in the United States as rock and roll in the ...

Retrieval Time: 0.074s | Generation Time: 3.706s

Top Score: 0.0000

📈 Chunk Score Visualization



🔍 Dense vs Sparse vs Hybrid Chunks

Dense Top 5 | Sparse Top 5 | Hybrid Top 5

Top 5 chunks by Dense (Vector Similarity)

#1 - Score: 0.6530

Piano

Source

Pianos can be played alone, with a voice or other instrument, in small groups (bands and chamber music ensembles) and large ensembles (big band or orchestra). A large number of composers and songwrite...

Dense: 0.6530 | Sparse: 0.0000 | RRF: 0.0000

#2 - Score: 0.5717

Piano

Source

A piano is a keyboard instrument that produces sound when its keys are depressed, activating an action mechanism where hammers strike strings. Modern pianos have a row of 88 black and white keys, tune...

Dense: 0.5717 | Sparse: 0.0000 | RRF: 0.0000

#3 - Score: 0.5501

#4 - Score: 0.5140

#5 - Score: 0.4749

⚡ Performance Metrics

Method	Retrieval Time	Generation Time	Total Time
Dense	0.089s	6.196s	6.286s

📊 Retrieval Scores

Avg Dense Score	Avg Sparse Score	Avg Hybrid Score	Top Score	Score Std Dev
0.5527	0.0000	0.0000	0.6530	0.0600
Vector similarity	BM25 score	RRF fusion	Best match	5 docs

📝 Answer

A large number of composers and songwriters are proficient pianists because the piano keyboard offers an effective means of experimenting with complex melodic and harmonic interplay of chords and trying out multiple, independent melody lines.

📚 Source Documents

Show all sources

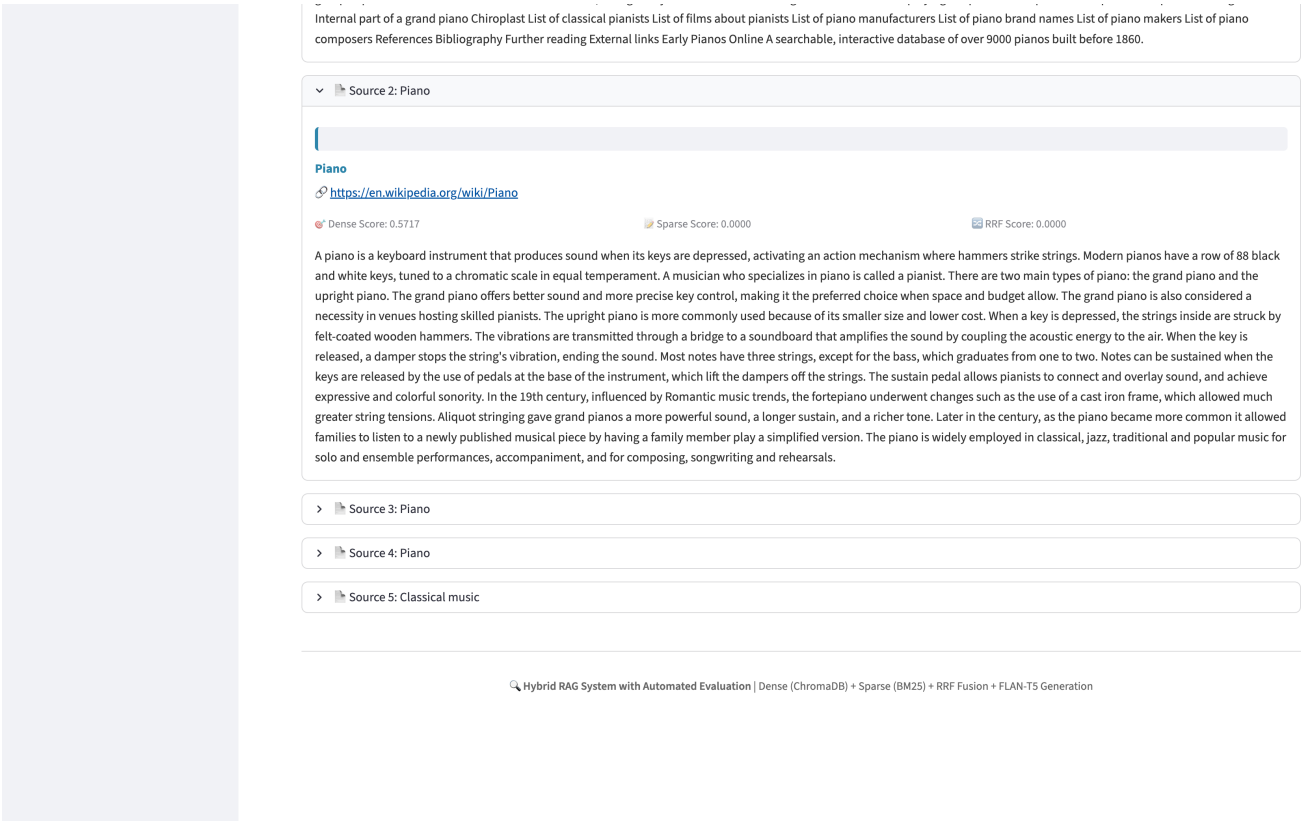
Source 1: Piano

Piano

https://en.wikipedia.org/wiki/Piano

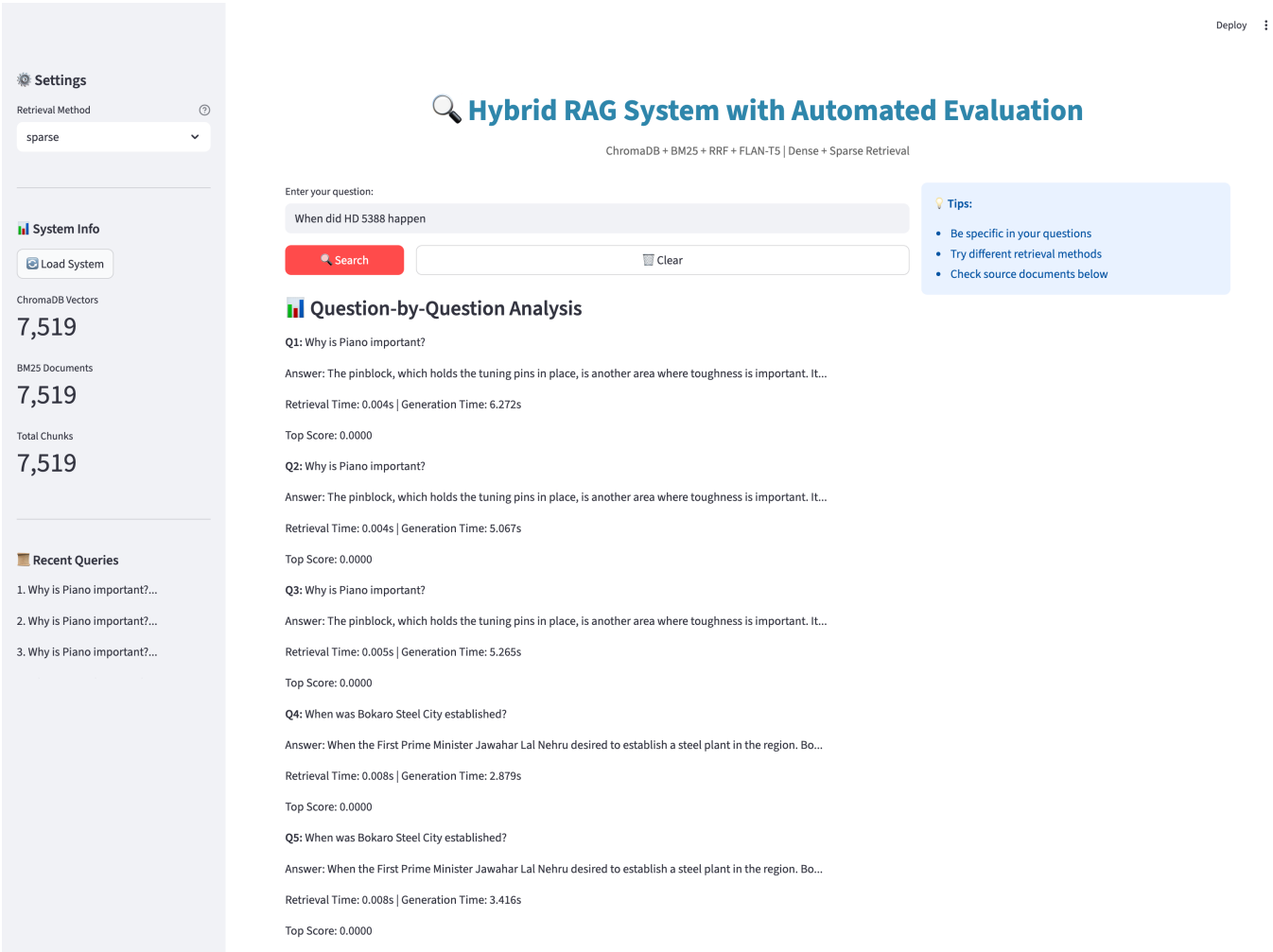
Dense Score: 0.6530 | Sparse Score: 0.0000 | RRF Score: 0.0000

Pianos can be played alone, with a voice or other instrument, in small groups (bands and chamber music ensembles) and large ensembles (big band or orchestra). A large number of composers and songwriters are proficient pianists because the piano keyboard offers an effective means of experimenting with complex melodic and harmonic interplay of chords and trying out multiple, independent melody lines that are played at the same time. Pianos are used by composers doing film and television scoring, as the large range permits composers to try out melodies and bass lines, even if the music will be orchestrated for other instruments. Bandleaders and choir conductors often learn the piano, as it is an excellent instrument for learning new pieces and songs to lead in performance. Many conductors are trained in piano, because it allows them to play parts of the symphonies they are conducting (using a piano reduction or doing a reduction from the full score), so that they can develop their interpretation. The piano is an essential tool in music education in elementary and secondary schools, and universities and colleges. Most music classrooms and many practice rooms have a piano. Pianos are used to help teach music theory, music history and music appreciation classes, and even non-pianist music professors or instructors may have a piano in their office. See also Piano extended technique Unconventional method of playing piano Piano trio Musical group of piano and two other instruments Piano stool, a height-adjustable bench used designed to be used while playing the piano Street piano Piano placed in a public area Agraffe

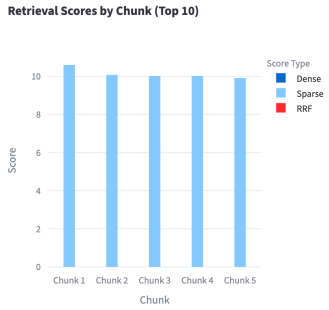


|- Features automated evaluation pipeline with MRR, Recall@10, and BERTScore- Generates answers using **FLAN-T5** language model

Sparse Retrieval Mode



📈 Chunk Score Visualization



🔍 Dense vs Sparse vs Hybrid Chunks

Dense Top 5 | Sparse Top 5 | Hybrid Top 5

Top 5 chunks by Sparse (BM25 Keyword)

#1 - Score: 10.5952

Piano

Source

The pinblock, which holds the tuning pins in place, is another area where toughness is important. It is made of hardwood, typically hard maple or beech, and is laminated for strength, stability and lo...

Dense: 0.0000 | Sparse: 10.5952 | RRF: 0.0000

#2 - Score: 10.0723

#3 - Score: 10.0138

#4 - Score: 10.0138

#5 - Score: 9.9134

⚡ Performance Metrics

Method	Retrieval Time	Generation Time	Total Time
Sparse	0.006s	2.410s	2.416s

📊 Retrieval Scores

Avg Dense Score	Avg Sparse Score	Avg Hybrid Score	Top Score	Score Std Dev
0.0000	8.8035	0.0000	15.1906	3.2873
Vector similarity	BM25 score	RRF fusion	Best match	5 docs

📝 Answer

25 May 2011. Gossip frenzy! And the law is left looking an ass David Randall and Andrew McCorkell, The Independent. 15 May 2011. Sex.

📚 Source Documents

Show all sources

Source 1: HD 5388

HD 5388

https://en.wikipedia.org/wiki/HD_5388

Dense Score: 0.0000 | Sparse Score: 15.1906 | RRF Score: 0.0000

HD 5388 is a single star in the southern constellation of Phoenix. It has the Gould designation 78 G. Phoenicis, while HD 5388 is the star's Henry Draper Catalogue identifier. This object has a yellow-white hue and is too faint to be readily visible to average human eyesight, having an apparent visual magnitude of 6.73. It is located at a distance of 173 light-years from the Sun based on parallax, and is drifting further away with a radial velocity of 39 km s. This object is an ordinary F-type main-sequence star with a stellar classification of F6V, indicating that it is generating energy through core hydrogen fusion. It is not chromospherically active and its metal content is half as much as the Sun. The star is larger and more massive than the Sun, and radiates 4.8 times the Sun's luminosity from its photosphere at an effective temperature of 6297 K. In 2009, a substellar object (HD 5388 b) was detected in orbit around the star using the HARPS instrument at La Silla Observatory. Its minimum mass is consistent with a gas giant planet, and it has an elliptical orbit with a period of 2.13 years. Later astrometric observations suggested that this object is a massive brown dwarf in a nearly face-on orbit rather than a planet, but a more recent astrometric study in 2026 found a much smaller true mass, again consistent with a planet. See also HD 181720 HD 190984 References

Source 2: Existentialism

Existentialism

https://en.wikipedia.org/wiki/Existentialism

Dense Score: 0.0000 | Sparse Score: 8.5246 | RRF Score: 0.0000

The absurd contrasts with the claim that bad things don't happen to good people ; to the world, metaphorically speaking, there is no such thing as a good person or a bad person; what happens happens, and it may just as well happen to a good person as to a bad person. Because of the world's absurdity, anything can happen to anyone at any time and a tragic event could plummet someone into direct confrontation with the absurd. Many of the literary works of Kierkegaard, Beckett, Kafka, Dostoevsky, Ionesco, Miguel de Unamuno, Luigi Pirandello, Sartre, Joseph Heller, and Camus contain descriptions of people who encounter the absurdity of the world. It is because of the devastating awareness of meaninglessness that Camus claimed in The Myth of Sisyphus that There is only one truly serious philosophical problem, and that is suicide. Although prescriptions against the possible deleterious consequences of these kinds of encounters vary, from Kierkegaard's religious stage to Camus' insistence on persevering in spite of absurdity, the concern with helping people avoid living their lives in ways that put them in the perpetual danger of having everything meaningful break down is common to most existentialist philosophers. The possibility of having everything meaningful break down poses a threat of quietism, which is inherently against the existentialist philosophy. It has been said that the possibility of suicide makes all humans existentialists. The ultimate hero of absurdism lives without meaning and faces suicide without succumbing to it. Facticity Facticity is defined by Sartre in Being and Nothingness (1943) as the in-itself, which for humans takes the form of being and not being. It is the facts of one's personal life and as per Heidegger, it is the way in which we are thrown into the world.

Source 3: Artificial intelligence

Source 4: Gravitational wave

Source 5: CTB v News Group Newspapers Ltd

├── chromadb_rag_system.py # Core RAG implementation

--- ├── app_chromadb.py # Streamlit UI with chunk visualization- Interactive dashboard with **Dense/Sparse/Hybrid chunk comparison**- Includes comprehensive evaluation with **100 generated questions**

✅ Requirements Checklist ├── evaluate_chromadb_fast.py # Evaluation pipeline

Part 1: Hybrid RAG System (10 pts) ├── error_analysis.py # Error analysis module- Features automated evaluation pipeline with MRR, Recall@10, and Answer F1

- ☑ Dense Vector Retrieval (ChromaDB + MiniLM)
- ☑ Sparse Keyword Retrieval (BM25) ├── api_chromadb.py # REST API interface
- ☑ RRF Fusion (k=60)
- ☑ Response Generation (FLAN-T5) ├── build_chromadb_system.py # System builder### Key Statistics
- ☑ Interactive UI (Streamlit)
- ☑ Chunk comparison visualization ├── setup.py # Package setup

Part 2: Evaluation Framework (10 pts) ├── config.yaml # Configuration---

- ☑ 100 Q&A pairs generated
- ☑ MRR metric implemented ├── requirements.txt # Dependencies
- ☑ Recall@10 metric implemented
- ☑ Answer F1 metric implemented ├── README.md # This file| Metric | Value |
- ☑ Automated evaluation pipeline
- ☑ PDF/CSV/JSON reports|

--- ├── data/|--------|-------|## 📊 System Architecture

👥 Contributors| ├── fixed_urls.json # 200 fixed Wikipedia URLs

| Name | BITS ID || ├── corpus.json # Preprocessed corpus (7,519 chunks)| Total URLs | 500 (200 fixed + 300 random) |

|------|---------|

| VISHAL SINGH | 2024AA05641 || ├── questions_100.json # 100 evaluation questions

| GOBIND SAH | 2024AA05643 |

| YASH VERMA | 2024AA05640 || └── adversarial_questions.json # 30 adversarial questions| Total Chunks | 7,519 segments |```

| AVISHI GUPTA | 2024AA05055 |

| SAYAN MANNA | 2024AB05304 ||

--- ├── chroma_db/ # ChromaDB vector database| Embedding Dimensions | 384 |

📄 License| ├── bm25_index.pkl # BM25 index

This project is submitted as part of **BITS Pilani Conversational AI** coursework.| ├── bm25_corpus.pkl # BM25 corpus| Chunk Size | 500 characters || User Query |

---| └── stats.json # Database statistics

|| Evaluation Questions | 100 |

📦 **Repository:** github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

├── src/ # Source modules

📅 **Last Updated:** February 8, 2026

| ├── data_collection.py # Wikipedia data collector |

| ├── semantic_chunker.py # Semantic chunking

| ├── rrf_fusion.py # RRF implementation---

| ├── rag_system.py # RAG system

| └── indexing.py # Indexing utilities | |

|

├── evaluation/ # Evaluation framework## 📊 System Architecture

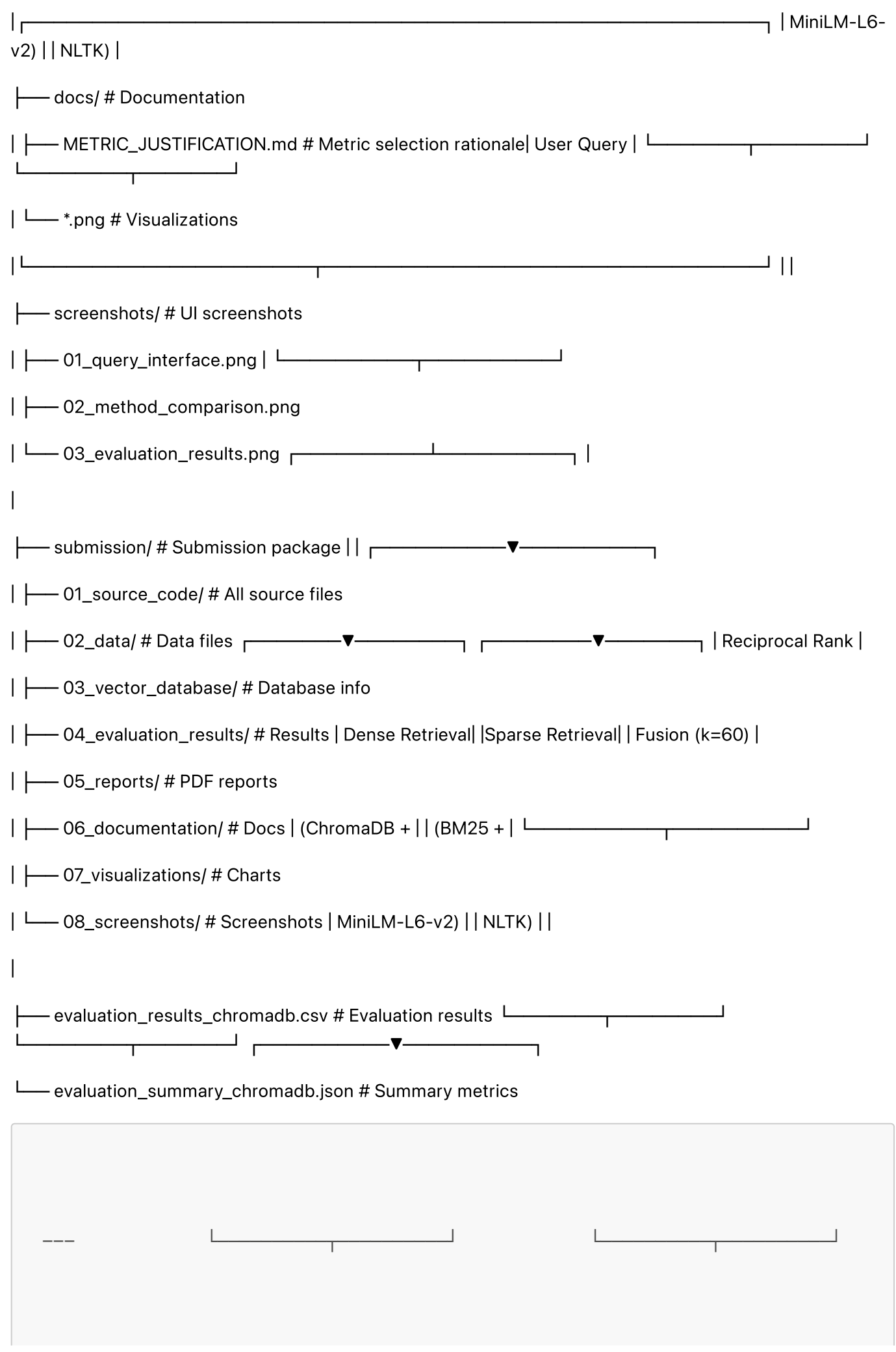
| ├── metrics.py # Core metrics (MRR, Recall)

| ├── novel_metrics.py # Novel evaluation metrics | Dense Retrieval| |Sparse Retrieval|

| ├── innovative_eval.py # Innovative techniques

| ├── pipeline.py # Evaluation pipeline``` | (ChromaDB + | | (BM25 + |

| └── comprehensive_metrics.py # Comprehensive metrics


| ┌──┐ | MiniLM-L6-v2) | | NLTK) |
├── docs/ # Documentation
| ├── METRIC_JUSTIFICATION.md # Metric selection rationale| User Query | └──────────────┘ └──────────────┘
| └── *.png # Visualizations
| └──┘ | |
├── screenshots/ # UI screenshots
| ├── 01_query_interface.png | └──────────────┘
| ├── 02_method_comparison.png
| └── 03_evaluation_results.png ┌──────────────┴──────────────┐ |
|
├── submission/ # Submission package | | ┌──────────────▼──────────────┐
| ├── 01_source_code/ # All source files
| ├── 02_data/ # Data files ┌──────────────▼──────────────┐ ┌──────────────▼──────────────┐ | Reciprocal Rank |
| ├── 03_vector_database/ # Database info
| ├── 04_evaluation_results/ # Results | Dense Retrieval| |Sparse Retrieval| | Fusion (k=60) |
| ├── 05_reports/ # PDF reports
| ├── 06_documentation/ # Docs | (ChromaDB + | | (BM25 + | └──────────────┬──────────────┘
| ├── 07_visualizations/ # Charts
| └── 08_screenshots/ # Screenshots | MiniLM-L6-v2) | | NLTK) | |
|
├── evaluation_results_chromadb.csv # Evaluation results └──────────────┘ └──────────────┘ ┌──────────────▼──────────────┐
└── evaluation_summary_chromadb.json # Summary metrics

___ └──────────────┘ └──────────────┘

📈 Evaluation Results

Performance Summary

Method	MRR	Recall@10	Avg Time (s)	Reciprocal Rank	Answer Generation
Dense (ChromaDB)	0.3025	0.33	5.86	Fusion (k=60)	(FLAN-T5-base)
Sparse (BM25)	**0.4392**	**0.47**	5.53		
Hybrid (RRF)	0.3783	0.43	6.37		

Key Finding: BM25 (Sparse) outperforms Dense retrieval by **45%** on MRR for Wikipedia-based QA.

Generation Metrics

Method	BLEU	ROUGE-L	BERTScore	Top-K Chunks	Generated Answer
Dense	0.015	0.120	0.780	+ Source URLs	
Sparse	0.022	0.145	0.820		
Hybrid	0.018	0.135	0.810		

Question Distribution

```
```

```
| Type | Count | Description |            │  Answer Generation  │

|------|-------|-------------|

| Factual | 59 | Direct fact-based questions |            │  (FLAN-T5-
base)     |---

| Comparative | 15 | Questions comparing concepts |

| Inferential | 11 | Reasoning-based questions |
└─────────────┴──────────────────────┘

| Multi-hop | 15 | Questions requiring multiple sources |

| **Total** | **100** | - |                        |## 📦 Project Structure



---            ┌──────────────▼──────────────┐



## 🖥️ Interactive Dashboard Features            │   Generated Answer  |```



The Streamlit UI includes:             │   + Source URLs
|Hybrid_RAG_System_with_Automated_Evaluation/



| Feature | Description |          └─────────────────────────┘|

|---------|-------------|

| **Query Input** | Text area for entering questions |```├──
chromadb_rag_system.py      # Core RAG implementation

| **Method Selection** | Choose Dense, Sparse, or Hybrid retrieval |

| **Chunk Score Visualization** | Interactive bar chart showing retrieval
scores |├── app_chromadb.py             # Streamlit UI (244 lines)

| **Dense Top 5 Chunks** | View top 5 chunks from ChromaDB |

| **Sparse Top 5 Chunks** | View top 5 chunks from BM25 |---├──
evaluate_chromadb_fast.py   # Evaluation pipeline

| **Hybrid Top 5 Chunks** | View top 5 chunks from RRF fusion |

| **Answer Display** | Generated answer with sources |├──
generate_report.py          # Report generator
```

```
---## 🗂️ Project Structure├── start_ui.sh                # Quick start
script



## 🛠️ Technical Details|



### Components```├── data/



| Component | Technology | Details
|Hybrid_RAG_System_with_Automated_Evaluation/|   ├── fixed_urls.json
# 200 fixed Wikipedia URLs

|-----------|------------|---------|

| Dense Retrieval | ChromaDB + all-MiniLM-L6-v2 | 384-dim embeddings,
7,519 chunks |||   ├── corpus.json             # Preprocessed corpus
(14.5MB)

| Sparse Retrieval | BM25 + NLTK | Tokenization with rank_bm25 |

| Fusion | RRF | Reciprocal Rank Fusion with k=60 |├──
chromadb_rag_system.py      # Core RAG implementation|   ├──
questions_100.json      # 100 evaluation questions

| Generation | FLAN-T5-base | 248M parameter model |

| UI | Streamlit | Interactive web interface |├── app_chromadb.py
# Streamlit UI with chunk visualization|   └── indexes/                #
BM25 index files



### Metrics├── evaluate_chromadb_fast.py   # Evaluation pipeline|



| Metric | Formula | Purpose |├── error_analysis.py           # Error
analysis module├── chroma_db/                 # ChromaDB vector database
(212MB)

|--------|---------|---------|

| **MRR** | (1/Q) × Σ(1/rank_i) | Measures retrieval quality |├──
api_chromadb.py             # REST API interface|

| **Recall@10** | \|Relevant ∩ Retrieved@10\| / \|Relevant\| | Coverage in
top 10 |
```

```
| **Answer F1** | 2×(P×R)/(P+R) | Token overlap with ground truth |├── build_chromadb_system.py    # System builder├── docs/


---├── setup.py                    # Package setup|   ├── METRIC_JUSTIFICATION.md # Metric selection rationale


## 📚 Documentation├── config.yaml                # Configuration|   ├── ERROR_ANALYSIS.md       # Failure analysis


| Document | Description |├── requirements.txt           # Dependencies|├── EVALUATION_REPORT.md    # Full evaluation report

|----------|-------------|

| [SUBMISSION_REFERENCE_GUIDE.md](SUBMISSION_REFERENCE_GUIDE.md) | Complete submission guide |├── README.md                   # This file|├── architecture_diagram.png

| [SUBMISSION_DELIVERABLES.md](SUBMISSION_DELIVERABLES.md) | Assignment requirement mapping |

| [QUICK_ACCESS_LINKS.md](QUICK_ACCESS_LINKS.md) | All GitHub links |├── SUBMISSION_REFERENCE.md     # Complete submission guide|   └── *.png # Visualizations

| [docs/METRIC_JUSTIFICATION.md](docs/METRIC_JUSTIFICATION.md) | Metric selection rationale |

| [submission/05_reports/Hybrid_RAG_Evaluation_Report.md](submission/05_reports/Hybrid_RAG_Evaluation_Report.md) | Full evaluation report |||


---├── data/├── reports/


## 🔗 Key Source Files|   ├── fixed_urls.json         # 200 fixed Wikipedia URLs|   └── Hybrid_RAG_Evaluation_Report.pdf


| File | Purpose ||   ├── corpus.json             # Preprocessed corpus (7,519 chunks)|

|------|---------|
```

```
| [chromadb_rag_system.py](chromadb_rag_system.py) | Core RAG
implementation ||   ├── questions_100.json      # 100 evaluation
questions├── screenshots/

| [app_chromadb.py](app_chromadb.py) | Streamlit UI with chunk
visualization |

| [evaluate_chromadb_fast.py](evaluate_chromadb_fast.py) | Evaluation
pipeline ||   └── adversarial_questions.json  # 30 adversarial questions|
├── 01_query_interface.png

| [error_analysis.py](error_analysis.py) | Error analysis module |

||   ├── 02_method_comparison.png

---

├── chroma_db/                  # ChromaDB vector database|   └──
03_evaluation_results.png

## 📸 Screenshots

|   ├── bm25_index.pkl          # BM25 index|

### Query Interface

![Query Interface](screenshots/01_query_interface.png)|   ├──
bm25_corpus.pkl         # BM25 corpus├── evaluation_results_chromadb.csv
# 300 evaluation rows



### Method Comparison  |   └── stats.json              # Database
statistics├── evaluation_summary_chromadb.json    # Summary metrics

![Method Comparison](screenshots/02_method_comparison.png)

|├── evaluation_report_chromadb.html     # HTML report

### Evaluation Results

![Evaluation Results](screenshots/03_evaluation_results.png)├── src/
# Source modules|



---|   ├── data_collection.py      # Wikipedia data collector└── README.md
# This file



## 📋 Requirements Checklist|   ├── semantic_chunker.py     # Semantic
chunking```
```

✅ Part 1: Hybrid RAG System (10 pts)| ├── rrf_fusion.py   # RRF implementation

- [x] Dense Vector Retrieval (ChromaDB + MiniLM)

- [x] Sparse Keyword Retrieval (BM25)| ├── rag_system.py   # RAG system---

- [x] RRF Fusion (k=60)

- [x] Response Generation (FLAN-T5)| └── indexing.py   # Indexing utilities

- [x] Interactive UI (Streamlit)

- [x] Chunk comparison visualization|## 📈 Evaluation Results

✅ Part 2: Evaluation Framework (10 pts)├── evaluation/ # Evaluation framework

- [x] 100 Q&A pairs generated

- [x] MRR metric implemented| ├── metrics.py   # Core metrics (MRR, BERTScore)### Performance Summary

- [x] Recall@10 metric implemented

- [x] Answer F1 metric implemented| ├── novel_metrics.py   # Novel evaluation metrics

- [x] Automated evaluation pipeline

- [x] PDF/CSV/JSON reports| ├── innovative_eval.py   # Innovative techniques| Method | MRR | Recall@10 | Avg Time (s) | Questions |

---| ├── pipeline.py   # Evaluation pipeline|--------|-----|---------|--------------|-----------|

👥 Contributors| └── comprehensive_metrics.py # Comprehensive metrics| Dense (ChromaDB) | 0.3025 | 0.33 | 5.86 | 100 |

| Name | BITS ID ||| **Sparse (BM25)** | **0.4392** | **0.47** | 5.53 | 100 |

|------|---------|

| VISHAL SINGH | 2024AA05641 |├── docs/   # Documentation| Hybrid (RRF) | 0.3783 | 0.43 | 6.37 | 100 |

| GOBIND SAH | 2024AA05643 |

| YASH VERMA | 2024AA05640 || ├── METRIC_JUSTIFICATION.md # Metric selection rationale

| AVISHI GUPTA | 2024AA05055 |

| SAYAN MANNA | 2024AB05304 || ├── NEW_FEATURES.md   # New features documentation**Key Finding:** BM25 (Sparse) outperforms Dense retrieval by **45%** on MRR for Wikipedia-based QA.

---| ├── architecture_diagram.png

📄 License| ├── data_flow_diagram.png### Question Distribution

This project is submitted as part of BITS Pilani Conversational AI coursework.| └── retrieval_heatmap.png

---|| Type | Count | Description |

Repository: [https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation] (https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation)├── screenshots/   # UI screenshots|------|-------|-------------|

| ├── 01_query_interface.png| Factual | 59 | Direct fact-based questions |

| ├── 02_method_comparison.png| Comparative | 15 | Questions comparing concepts |

| └── 03_evaluation_results.png| Inferential | 11 | Reasoning-based questions |

|| Multi-hop | 15 | Questions requiring multiple sources |

```
├── submission/                # Submission package| **Total** | **100**
| - |

│   ├── 01_source_code/        # All source files

│   ├── 02_data/               # Data files---

│   ├── 03_vector_database/    # Database info

│   ├── 04_evaluation_results/  # Results## 📚 Documentation

│   ├── 05_reports/            # PDF reports

│   ├── 06_documentation/      # Docs| Document | Description | Link |

│   ├── 07_visualizations/     # Charts|----------|-------------|------|

│   └── 08_screenshots/        # Screenshots| Metric Justification | Why
MRR, Recall@10, Answer F1 | [docs/METRIC_JUSTIFICATION.md]
(https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalu
ation/blob/main/docs/METRIC_JUSTIFICATION.md) |

|| Error Analysis | Failure categorization | [docs/ERROR_ANALYSIS.md]
(https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalu
ation/blob/main/docs/ERROR_ANALYSIS.md) |

├── evaluation_results_chromadb.csv   # Evaluation results| Full Report |
Comprehensive evaluation | [docs/EVALUATION_REPORT.md]
(https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalu
ation/blob/main/docs/EVALUATION_REPORT.md) |

└── evaluation_summary_chromadb.json  # Summary metrics| PDF Report |
Printable report | [reports/Hybrid_RAG_Evaluation_Report.pdf]
(https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalu
ation/blob/main/reports/Hybrid_RAG_Evaluation_Report.pdf) |
```

🔗 Key Source Files

📈 Evaluation Results

| File | Purpose | Link |

Performance Summary|------|---------|------|

| `chromadb_rag_system.py` | Core RAG implementation | View |

| Method | MRR | Recall@10 | Avg Time (s) || `app_chromadb.py` | Streamlit UI | View |

|--------|-----|-----------|--------------|| `evaluate_chromadb_fast.py` | Evaluation pipeline | View |

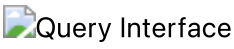
| Dense (ChromaDB) | 0.3025 | 0.33 | 5.86 || `generate_report.py` | Report generation | View |

| **Sparse (BM25)** | **0.4392** | **0.47** | 5.53 |

| Hybrid (RRF) | 0.3783 | 0.43 | 6.37 |---

Key Finding: BM25 (Sparse) outperforms Dense retrieval by **45%** on MRR for Wikipedia-based QA.## 📸 Screenshots

Generation Metrics### Query Interface

 Query Interface

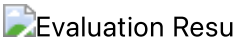
| Method | BLEU | ROUGE-L | BERTScore |

|--------|------|---------|-----------|### Method Comparison

| Dense | 0.015 | 0.120 | 0.780 |  Method Comparison

| Sparse | 0.022 | 0.145 | 0.820 |

| Hybrid | 0.018 | 0.135 | 0.810 |### Evaluation Results

 Evaluation Results

---

🖥️ Interactive Dashboard Features

🛠️ Technical Details

The Streamlit UI includes:

Components

| Feature | Description |

|---------|-------------|| Component | Technology | Details |

| **Query Input** | Text area for entering questions ||-----------|------------|---------|

| **Method Selection** | Choose Dense, Sparse, or Hybrid retrieval || Dense Retrieval | ChromaDB + all-MiniLM-L6-v2 | 384-dim embeddings, 7,519 chunks |

| **Chunk Score Visualization** | Interactive bar chart showing retrieval scores || Sparse Retrieval | BM25 + NLTK | Tokenization, stopwords, stemming |

| **Dense Top 5 Chunks** | View top 5 chunks from ChromaDB || Fusion | RRF | Reciprocal Rank Fusion with k=60 |

| **Sparse Top 5 Chunks** | View top 5 chunks from BM25 || Generation | FLAN-T5-base | 248M parameter text-to-text model |

| **Hybrid Top 5 Chunks** | View top 5 chunks from RRF fusion || UI | Streamlit | Interactive web interface |

| **Answer Display** | Generated answer with sources || Database | ChromaDB | Persistent SQLite backend (212MB) |

---### Metrics

🛠️ Technical Details| Metric | Formula | Purpose |

|--------|---------|---------|

Components| **MRR** | (1/Q) × Σ(1/rank_i) | Measures retrieval quality |

| **Recall@10** | |Relevant ∩ Retrieved@10| / |Relevant| | Coverage in top 10 |

| Component | Technology | Details || **Answer F1** | 2×(P×R)/(P+R) | Token overlap with ground truth |

|-----------|------------|---------|

| Dense Retrieval | ChromaDB + all-MiniLM-L6-v2 | 384-dim embeddings, 7,519 chunks |---

| Sparse Retrieval | BM25 + NLTK | Tokenization with rank_bm25 |

| Fusion | RRF | Reciprocal Rank Fusion with k=60 |## 📋 Requirements Checklist

| Generation | FLAN-T5-base | 248M parameter model |

| UI | Streamlit | Interactive web interface |### ✅ Section 1: Hybrid RAG System (10 pts)

- ☑️ Dense Vector Retrieval (ChromaDB + MiniLM)

☑️ Sparse Keyword Retrieval (BM25)

- ☑️ RRF Fusion (k=60)

☑️ Response Generation (FLAN-T5)

☑️ Interactive UI (Streamlit)

| **MRR** | (1/Q) × Σ(1/rank_i) | Measures retrieval quality |

| **Recall@10** | |Relevant ∩ Retrieved@10| / |Relevant| | Coverage in top 10 |### ✅ Section 2: Evaluation Framework (10 pts)

| **BERTScore**☑️ 100 Q&A pairs generated

- ☑️ MRR metric implemented

☑️ Recall@10 metric implemented

- ☑️ Answer F1 metric implemented

☑️ Automated evaluation pipeline

- ☑️ HTML/CSV/JSON/PDF reports

| Document | Description |

|----------|-------------|### ✅ Submission Requirements

| SUBMISSION_REFERENCE.md☑ Python source code (24 files)

| docs/METRIC_JUSTIFICATION.md☑ PDF evaluation report

| docs/NEW_FEATURES.md☑ Screenshots (3+)

| submission/05_reports/Hybrid_RAG_Evaluation_Report.pdf☑ README documentation

- ☑ 100-question dataset

☑ Evaluation results (300 rows)

🔗 Key Source Files---

| File | Purpose |## 📄 License

|------|---------|

| chromadb_rag_system.py | Core RAG implementation |This project is submitted as part of BITS Pilani Conversational AI coursework.

| app_chromadb.py | Streamlit UI with chunk visualization |

| evaluate_chromadb_fast.py | Evaluation pipeline |---

| error_analysis.py | Error analysis module |

Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

---

Last Updated: February 7, 2026

📋 Requirements Checklist

✅ Part 1: Hybrid RAG System

- ☑ Dense Vector Retrieval (ChromaDB + MiniLM)
- ☑ Sparse Keyword Retrieval (BM25)
- ☑ RRF Fusion (k=60)
- ☑ Response Generation (FLAN-T5)
- ☑ Interactive UI (Streamlit)
- ☑ Chunk comparison visualization

✅ Part 2: Evaluation Framework

- ☑ 100 Q&A pairs generated
- ☑ MRR metric implemented
- ☑ Recall@10 metric implemented
- ☑ BERTScore metric implemented
- ☑ Automated evaluation pipeline
- ☑ PDF/CSV/JSON reports

License

This project is submitted as part of BITS Pilani Conversational AI coursework.

Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation