Name: Fenil Vadher

En Roll no: 92200133023

Subject: Capstone Project

System Design and Architecture

1. Modular Design

The proposed **Multimodal Movie Script Search System** is designed using a **modular architecture** to ensure maintainability, extensibility, and scalability. The system is divided into the following major modules:

1. **User Interface (UI) Module**
   - Provides an intuitive front-end for users to query the system.
   - Supports **text, image, and dialogue input queries**.
   - Displays ranked search results with **scene previews and metadata**.
2. **Query Processing and Preprocessing Module**
   - Normalizes and tokenizes user input.
   - Converts queries (text, image, audio) into embeddings compatible with retrieval models.
   - Applies language processing (stemming, lemmatization) for textual queries.
3. **Multimodal Embedding and Retrieval Module**
   - Core AI engine powered by **Vid2Seq, BLIP-2, mPLUG, GIT2, Sky, SPtPT**.
   - Generates **unified embeddings** for movie scripts, dialogues, and visual scenes.

- Implements **semantic similarity search** to retrieve contextually relevant results.

4. **Database and Indexing Module**
   - Stores structured movie scripts, dialogue transcripts, and scene metadata.
   - Uses **vector databases (e.g., Pinecone, FAISS, or Milvus)** for efficient embedding retrieval.

5. **Backend API and Orchestration Module**
   - Exposes RESTful APIs/GraphQL endpoints for queries.
   - Orchestrates communication between front-end, embedding models, and databases.
   - Ensures secure authentication and access control.

6. **Evaluation and Analytics Module**
   - Integrates evaluation metrics: **BLEU, METEOR, CIDEr, ROUGE-L, CLIP-Similarity, Precision, Recall**.
   - Provides reports for model performance and retrieval quality.
   - Logs user interactions to refine recommendation strategies.

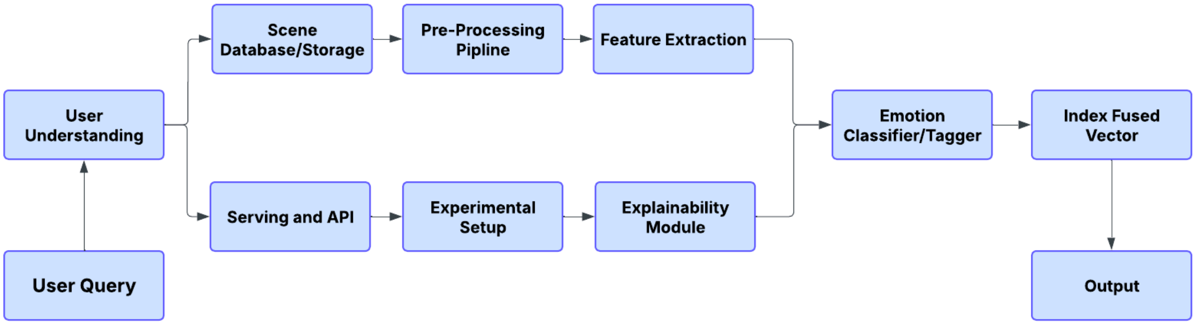
Justification of Modularity:

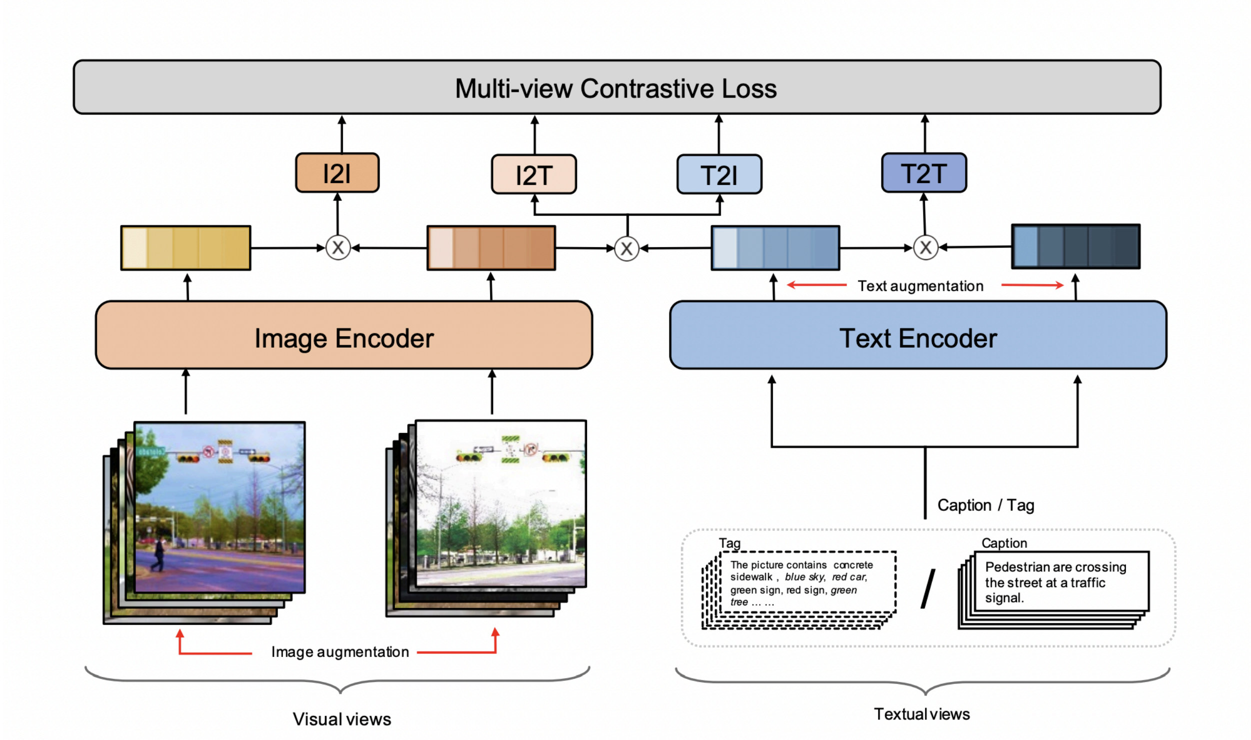
- Each module is **independent yet loosely coupled**, enabling upgrades without disrupting the entire system.

- Supports **reusability** (e.g., embedding module can be reused for other multimedia retrieval tasks).

- Facilitates **parallel development**, improving project efficiency.

2. System Architecture Diagram




Multi-view Contrastive Loss
I2I
I2T
T2I
T2T
X
X
X
Text augmentation
Image Encoder
Text Encoder
Caption / Tag
Tag
The picture contains concrete sidewalk , blue sky, red car, green sign, red sign, green tree … …
/
Caption
Pedestrian are crossing the street at a traffic signal.
Image augmentation
Visual views
Textual views

3. Technology Stack

Layer/Module	Technology	Justification
Frontend (UI)	React.js, Tailwind CSS	Provides responsive, modular UI with reusability.
Backend API	Node.js (Express) / FastAPI (Python)	Lightweight, high-performance API framework for handling queries.
AI Models	Vid2Seq, BLIP-2, mPLUG, GIT2, Sky, SPtPT	Pre-trained multimodal transformers for embedding and retrieval.
Text Processing	Hugging Face Transformers, SpaCy	Robust NLP processing and embeddings.
Vector Database	FAISS / Pinecone / Milvus	High-performance similarity search for embeddings.

Justification:

- React.js + Node.js/FastAPI → ensures low latency and modern web app support.
- Vector DB (FAISS/Milvus) → optimized for nearest-neighbor search on embeddings.

4. Scalability Planning

Model Scalability

- Deploy AI models on **GPU-enabled cloud instances (AWS SageMaker, GCP Vertex AI)**.
- Use **batch inference** and **model quantization** to reduce compute costs.
- As demand grows, implement **distributed inference frameworks (Ray, Horovod)**.

Cost and Reliability Considerations

- **Auto-scaling policies** to minimize idle resource cost.
- **Spot instances** for low-cost compute when possible.
- **Redundancy and failover strategies** to ensure uptime.