****

**Superior University Lahore**

**Programming for AI LAB**

**Task 11**

*Department of Software Engineering*

*Faculty of Computer Science & Information Technology*

*The Superior University, Lahore*

# **LangChain**

* **Framework for LLM Applications**: LangChain is an open-source framework designed to simplify every stage of building applications powered by large language models (LLMs), providing standardized components for prompts, chains, agents, memory, and integrations
* **Modular Design**: Offers reusable modules (e.g., prompt templates, output parsers, retrievers) that let developers stitch together complex workflows without reinventing common LLM-centric patterns
* **Focus on Orchestration**: Emphasizes “chains” (sequences of calls) and “agents” (LLM-driven decision-making loops) to automate multi-step reasoning or tool use

# **RAG (Retrieval-Augmented Generation)**

* **Hybrid Retrieval + Generation**: RAG enhances LLM outputs by first retrieving relevant external documents or knowledge from a corpus, then conditioning generation on that factual context.
* **Improves Accuracy & Consistency**: By grounding responses in up-to-date or domain-specific data, RAG mitigates hallucinations common in standalone LLM inference.
* **Flexible Pipelines**: Retrieval can use simple keyword search, vector similarity, or more advanced ranking models before LLM fusion.

# **LLMs (Large Language Models)**

* **Definition**: LLMs are deep neural networks with billions of parameters, trained via self-supervised learning on massive text corpora to perform tasks like text generation, completion, and classification.
* **Transformer Backbone**: Almost all modern LLMs use transformer architectures (attention-based encoders/decoders) for scalable, parallelizable training and inference.
* **General-Purpose Foundation**: Serve as “foundation models” for downstream fine-tuning, in-context learning, or RAG pipelines across domains.

# **FAISS (Facebook AI Similarity Search)**

* **Similarity Search Library**: FAISS is a C++ (with Python bindings) library for efficient nearest-neighbor search and clustering of dense vector embedding at billion-scale datasets.
* **High Performance**: Implements CPU and GPU indexes (e.g., IVF, HNSW) optimized for fast approximate search and memory efficiency.
* **Common in RAG Pipelines**: Frequently used to retrieve candidate documents based on embedding similarity before LLM fusion.

# **Vector (Embedding)**

* **Numerical Representations**: Vectors are fixed-length lists of numbers encoding semantics of words, sentences, images, or other data so that similarity corresponds to semantic relatedness.
* **Dimensionality & Space**: Typical embedding sizes range from hundreds to thousands of dimensions; nearest-neighbor operations exploit geometric properties in this space.
* **Foundation for Search & Clustering**: Core to similarity search, clustering, recommendation, and many downstream AI tasks.

# **VectorDB (Vector Database / Vector Store)**

* **Specialized Storage**: Databases optimized to store, index, and retrieve vector embedding efficiently, often supporting CRUD, metadata filtering, and scalable serving.
* **ANN Algorithms**: Employ Approximate Nearest Neighbor techniques (e.g., HNSW, PQ) to balance retrieval speed and accuracy.
* **Integrates with RAG**: Acts as the back end for retrieving context embedding for RAG workflows.

# **Generative AI**

* **Subfield of AI**: Involves models trained to generate new data (text, images, audio, video) that emulate the distribution of their training set.
* **Model Types**: Encompasses autoregressive LLMs, diffusion models, VAEs, GANs, and other generative architectures.
* **Applications**: Content creation, data augmentation, simulation, creative industries, and more.

# **GANs (Generative Adversarial Networks)**

* **Adversarial Dual-Network**: Consist of a Generator (creates samples) and a Discriminator (distinguishes real vs. fake); they train in a zero-sum game to improve generation quality iteratively.
* **Unsupervised Learning**: Can learn to generate high-fidelity images or other data without labeled examples, relying on adversarial feedback.
* **Use Cases**: Image synthesis, style transfer, super-resolution, data augmentation, and domain adaptation