Agentic AI:
- Agent: An autonomous program that observes the environment, reasons, and takes actions toward a goal.
- Agentic AI: A design paradigm where AI systems are built as goal-directed agents able to plan, act, and learn over time with minimal human intervention.
- Reactive Agent: An agent that chooses actions based only on the current observation, without internal planning or memory.
- Planning Agent: An agent that builds and executes a multi-step plan before acting, often using search or chain-of-thought reasoning.
- Multi-Agent System: A collection of agents that collaborate or compete, communicating via messages or shared state.
- Orchestration: Coordinating multiple agents or tools, deciding which one acts next and passing context between them.
- Tool Calling: Mechanism that lets an agent invoke external functions/APIs to extend its capabilities beyond pure language.
- Observation: Structured result returned to an agent after it calls a tool or queries data, used for subsequent reasoning.
- Memory: Persistent store (short- or long-term) that an agent can read/write, enabling context retention across steps.
- Guardrails: Policies or code that constrain an agent’s actions or outputs to enforce safety, compliance, or budget limits.
- Autonomy: Degree to which an agent can operate without human oversight, balancing initiative with safe boundaries.
- Feedback Loop: Iterative cycle where an agent’s actions are evaluated and the outcome is used to refine future behavior.

Generative AI:
- Generative AI: Algorithms that learn data distributions and create new content—text, images, audio, code—that resembles the training data.
- Large Language Model (LLM): A transformer-based neural network with billions of parameters trained on vast text corpora to predict the next token.
- Diffusion Model: Generative model that learns to remove noise step-by-step, enabling image, audio, or 3-D synthesis.
- Transformer: Deep learning architecture using self-attention to model token relationships; foundational to modern LLMs.
- Fine-Tuning: Additional training of a pre-trained model on domain-specific data to specialize its outputs.
- Prompt: Input text (plus optional system instructions) used to steer a generative model toward a desired completion.
- Token: Smallest unit of text (sub-word or character) processed by an LLM; each forward pass predicts the next token distribution.
- Temperature: Sampling hyper-parameter controlling randomness; lower values yield deterministic, higher values more creative outputs.
- Top-p (Nucleus) Sampling: Method that samples from the smallest token set whose cumulative probability ≥ p, trading diversity for coherence.
- Embedding: Dense vector representation of text capturing semantic meaning, used for search, clustering, and retrieval.
- Vector Store: Database optimized for similarity search over embeddings, supporting nearest-neighbor queries.
- Latent Space: High-dimensional feature space in which generative models learn a compressed representation of the data.
- RLHF: Reinforcement Learning from Human Feedback—technique aligning model outputs with human preferences.
- Hallucination: Confident response from a model that is factually incorrect or unsupported by the provided context.

System Architecture:
- Microservice: Small, independently deployable service that owns a single business capability and communicates via APIs.
- Monolith: Single deployable unit containing many tightly coupled components; opposite of microservice style.
- API Gateway: Entry point that routes, authenticates, and rate-limits requests to backend services.
- Message Queue: Asynchronous transport that decouples producers and consumers, enabling buffering and retry.
- Event-Driven Architecture: Pattern where services react to events rather than direct calls, improving scalability and responsiveness.
- Caching Layer: Low-latency store (e.g., Redis) used to avoid recomputation or database hits for hot data.
- Stateless Service: Service that keeps no session between requests, enabling easy horizontal scaling.
- Load Balancer: Component that distributes incoming traffic across multiple instances to maximize throughput and resilience.
- CI/CD: Continuous Integration and Continuous Delivery/Deployment pipelines automating build, test, and release.
- Infrastructure as Code: Declarative templates (e.g., Terraform, CloudFormation) that version-control cloud resources.

- Drift Detection: Monitoring technique that flags when data or model performance deviates significantly from training conditions.

LangChain:
- Chain: Composable sequence of calls—prompts, LLMs, tools—forming a higher-level task.
- Runnable: Unified interface in LangChain 0.1+ representing an object with .invoke, .stream, and .batch methods.
- LCEL: LangChain Expression Language, a DSL that lets you pipe components together (`prompt | model | parser`).
- PromptTemplate: Parameterized prompt with placeholders that can be filled at runtime.
- Retriever: Component that performs similarity search over documents and returns relevant chunks.
- VectorStore: LangChain wrapper around a vector database (e.g., PGVector, Pinecone, FAISS).
- Tool: LangChain abstraction for an external function callable by an agent through natural-language instructions.
- AgentExecutor: High-level loop that lets an LLM choose tools, observe results, and decide next actions.
- Memory Module: Interface that stores and retrieves conversational context for stateful interactions.
- OutputParser: Component that converts raw LLM output into structured Python objects.
- Callback: Hook fired during chain execution to log events or stream tokens.
- LangServe: FastAPI-based packaging of chains/agents into deployable REST or gRPC endpoints.
- LangGraph: State-machine framework built on LangChain for orchestrating complex, asynchronous agent workflows.
- DocumentLoader: Utility that reads data from files, URLs, or APIs into LangChain Document objects.
- TextSplitter: Algorithm that chunks documents into manageable sizes with configurable overlap.