MODULE 3 · STUDY NOTES

Introducing Generative AI

Foundation models, tokens and embeddings, output risks, the three AWS generative AI services, and choosing use cases

Generative AI moves machine learning from "predict a label" to "create the answer." At its heart sit foundation models — enormous pre-trained neural networks that generate text, images, audio, even protein structures, from a plain-language prompt. This module builds the working vocabulary: what FMs and LLMs are, how tokens, embeddings, and vectors turn language into math, and why every output is a probability-driven prediction — with the hallucination, toxicity, bias, and intellectual-property risks that follow. It then covers the levers that steer results (inference parameters, prompt design, additional data, safeguards), maps the three primary AWS services for generative AI — Amazon SageMaker AI, Amazon Bedrock, and Amazon Q — and closes with the discipline of working backward from a business problem to pick use cases, plus PartyRock, a free playground for experimenting.

LEARNING OBJECTIVES

- Describe key characteristics of foundation models (FMs)
- Define a prompt in the context of generative AI
- Describe task types for large language models (LLMs)
- Explain the role of tokens, embeddings, and vectors in LLMs
- Recognize the potential business value of generative AI
- Summarize key challenges of using generative AI, and discuss how to mitigate them
- Identify methods for using additional data to influence the outputs of an FM
- Describe three primary AWS services for generative AI
- Identify potential use cases for generative AI

3.1 Foundation models

The models behind generative AI are **foundation models (FMs)**: deep learning models based on complex neural networks, trained on massive amounts of data through extensive training. What distinguishes an FM from other deep learning models is the degree to which it produces **generalized responses** — you can ask an FM questions it was never specifically designed to answer, and it generates output far more sophisticated than other programming methods. That generality is what makes FMs powerful across a broad array of applications.

Even so, each model is typically trained — and architecturally designed — to **excel at a specific category of output**. The common buckets are language, image, audio, and specialized domains such as biomedical; **multimodal models** combine data types (text, images, sound) for a more comprehensive understanding — for example, analyzing a video by understanding the spoken words, recognizing the objects on screen, and reading any text that appears. The categories are neither definitive nor exhaustive, and models in every category can accept **natural-language inputs** even when their outputs take other forms.

TABLE 3.1 Common FM categories, uses, and example prompts

CATEGORY	GENERATES	EXAMPLE USES	EXAMPLE PROMPT
Language	Text-based responses	Virtual assistance, chat-based assistants	"Summarize distinctions between dogs and cats."
Image	High-resolution images	Product design, marketing	"Generate an image of a black puppy with large ears and a white kitten playing with a ball."
Audio	Music, sounds, speech	Game sound effects, synthesized speech	"Generate the sound of a joyful puppy barking."
Biomedical	Protein or molecular structures	Therapeutics design, new drug discovery	"Predict the three-dimensional structure of the canine olfactory receptor protein OR5B17."

Some organizations build their own FMs for customized uses, but many **pre-trained** foundation models are publicly available across all of these categories. Using an existing FM removes the need for your own extensive data collection, model training, and infrastructure setup — which **reduces startup costs and increases the speed of adoption**, and makes generative AI attractive for tasks that would have been challenging or cost-prohibitive to build internally.

To generate output from an existing FM, you give it an instruction called a **prompt** — an instruction or question written in natural language that requests the model to perform a specific task. Framed in machine-learning terms, the prompt is the equivalent of the *new data* you hand a trained model to make an **inference**. The generated content is therefore still a **prediction**: the model infers what the answer should be from what it has learned, and the quality of the answer depends not only on the training data and the model but also on **how well you phrase the question**. Finding the optimal way to write a prompt is a big part of designing a reliable generative AI solution.

LLMS – FOUNDATION MODELS FOR TEXT

- **Large language models (LLMs)** are FMs pre-trained extensively on massive text datasets — trillions of words.
- They capture intricate language patterns and relationships, so they can generate a broad variety of text content.
- LLMs specialize in natural language processing and are the FM type this module uses for its examples.

TABLE 3.2 Common LLM task types

TASK TYPE	WHAT YOU ASK THE MODEL TO DO	EXAMPLE PROMPT
Text generation	Write new text-based output	"Write a 300-word introduction for a pet-adoption website on the benefits and challenges of owning a pet."
Question & answer	Answer a specific question	"What are the 5 most popular types of pets globally?"
Summarization	Condense existing content	"Summarize distinctions between dogs and cats based on the attached article."
Text extraction	Pull specific information out of unstructured sources (reports, customer emails) into a structured format	"Extract the pet names, breeds, and notable characteristics or behaviors described, presented concisely."
Conversation	Hold a back-and-forth exchange — the model **retains the thread**	"What are the 10 most popular dog breeds?" … "What are their sizes?" — the model knows you still mean dogs
Code generation	Write or audit development code	"Write a Python function to post data about dogs available for adoption from a MySQL database to a website."

> **COMMON PITFALL**
>
> Generated content is a **prediction**, not a retrieved fact. The model infers what the answer *should* be based on what it learned — which is exactly why phrasing matters and why outputs need validation before you rely on them.

3.2 How foundation models work

Even among models trained for the same kind of task, each model's **architecture** (its underlying design) and training approach is distinct; the choice depends on the specific task, the available data, and the computational resources. Four common approaches underpin popular generative AI models, and the AI Practitioner exam may ask you to recognize their **distinguishing features and typical use cases**. Keep in mind that architectures evolve quickly, are often used in combination, and get adapted across domains — designs that began in natural language processing have been reworked for images, and vice versa.

TABLE 3.3 Common approaches to building generative AI models

ARCHITECTURE	HOW IT GENERATES	TYPICAL STRENGTH
Diffusion model	Learns to generate new data by **adding noise** to a clean input, then learning to **reverse the process** to remove the noise	High-quality images
Generative adversarial network (GAN)	Two neural networks trained competitively: the **generator** creates data samples meant to fool the **discriminator**, which is trained to classify input as real or fake	Realistic data samples
Variational autoencoder (VAE)	Learns a **latent representation** — essential information compressed into a lower-dimensional *latent space* for efficient processing — and generates new samples resembling the original	Efficient generation from compressed representations
Transformer	Processes sequences **in parallel** and pays **attention** to different parts of the input simultaneously — the feature that distinguishes it from earlier neural networks	Modern LLMs

You don't need the detailed math behind these models, but three concepts recur throughout the course — and some of them affect costs: **tokenization, embeddings, and vectors**. Tokenization breaks input text down into smaller units called **tokens** — words, subwords, or even individual characters depending on the tokenization strategy — converting raw text into a format the model can easily process. During **embedding**, each token is converted into a numerical representation called a **vector** (you will hear *embeddings* and *vectors* used interchangeably).

WHAT EMBEDDINGS CAPTURE

- Similar tokens get similar vectors: *cat* sits closer to *kitten* than to *puppy*.
- Two-dimensional diagrams are only illustrations — real embeddings are **dense vectors with hundreds or even thousands of dimensions**.
- The vectors feed the model's neural network during training; the model learns relationships between words and meanings, enabling language understanding, generation, and translation.
- These concepts return later in the course with retrieval-augmented techniques, vector databases, and embedding models.

WORKED EXAMPLE – PREDICTING WHAT PROBABLY COMES NEXT

"The cat jumped onto the …" You'd guess *couch* or *table* — *bowling ball* is possible but far less probable. Context shifts the odds: a **sleepy** cat nudges you toward *couch*; a **curious** cat leans *table*; add *a plate of fish* and *table* is nearly certain; and in a story about performing cats balancing on round objects on stage, *bowling ball* wins. LLMs do exactly this at much larger scale — they use the patterns and relationships they learned across words, sequences, and context to predict what should come next. Decide how creative the model should be for your use case, and use the available tools to guide its output.

Because FM answers are **probability-based predictions built from training data**, generated output carries risks beyond those of traditional AI. Four challenges recur, each with a mitigation posture:

TABLE 3.4 Challenges of FM-generated output and mitigations

CHALLENGE	WHAT CAN HAPPEN	MITIGATION
Hallucinations	Results look correct but are **verifiably untrue** — e.g., LLMs inventing nonexistent scientific citations	Balance creativity vs. accuracy for the use case; incorporate reviews to validate expected results
Toxicity	Offensive or inappropriate content; the line between restriction and censorship is murky, context- and culture-dependent, and subtly worded offense is hard to detect	Develop thoughtful guidelines and implement **guardrails**; define what is acceptable per audience (a vulgar quote may pass in journalism, never in a children's book)
Bias	Societal biases in training data are perpetuated or amplified — job descriptions that discourage certain groups, career advice reflecting gender stereotypes	Diverse, representative training data; careful model design; continuous monitoring and auditing; promote fairness **without overcorrecting** or introducing new bias
Illegality (IP)	Output uses or closely imitates protected text, code, or imagery — e.g., imitating a celebrity voice without permission	Set boundaries and incorporate **audit mechanisms**; expect technology, policy, and legal mechanisms to evolve together

Most of these problems start with the **data the model was trained on**, so careful selection and curation of training data is critical when building an FM. But you will usually be using an *existing* FM **without access to its training data** — so your solution must employ approaches that reduce the risk. Tolerance is use-case dependent: creative storytelling forgives a hallucination; a legal briefing does not.

The first steering lever is **inference parameters**, which individual models expose to influence responses. Restrictive settings make output **more deterministic** — conservative, consistent, more likely accurate — right for applications like answering human-resources questions. Looser settings make output **less deterministic**, giving the model more room for creativity — right for storytelling or artwork.

> **COMMON PITFALL**
>
> Keep the direction straight: **more deterministic = more restrictive = more accurate**; less deterministic = more creative. Exam wording flips this on purpose.

If you have tried several FMs and tuned parameters but still can't get the output you want, you do **not** need to build your own foundation model yet. Three strategies feed an existing FM **additional data** relevant to your goal, in rising order of complexity — and generally, higher complexity means more resources and higher cost:

TABLE 3.5 Using additional data to influence FM outputs

STRATEGY	WHAT YOU DO	CHILD ANALOGY	COMPLEXITY
Enrich the prompt	Add descriptive words, detailed instructions to follow, example outputs, or reference material	A worksheet with information and instructions	Least
External knowledge base	Let the model look up relevant information before answering — e.g., internal policy documents behind an HR question-answering app	An open-book test	Middle
Additional training	Customize the model with further training on a **smaller, task-specific dataset** — e.g., a medical dataset to sharpen medical answers	A tutor for the subject that matters	Most

FOUR KEY COMPONENTS FOR OPTIMAL OUTPUTS

- **Model choice** — start with a model whose training data and architecture suit the task; use inference parameters to optimize outputs.
- **Prompt design** — descriptive, thoughtful prompts guide the model toward the desired kind of answer; iterate to optimize.
- **Data quality** — additional data makes responses relevant to a specific domain; protect data across the solution's lifecycle.
- **Safeguards** — follow responsible AI practices; use tools that keep undesired content out of inputs and outputs; incorporate human reviews and robust auditing.

3.3 AWS generative AI services

Three AWS services span the continuum from *using* generative AI to *building models yourself*, from less to more complex. With **Amazon Q Business** you can use prompts or reference a knowledge base. **Amazon Bedrock** adds customizing a model with your own data. **Amazon SageMaker AI** adds building your own model. All three are **managed services** — you spend effort on the business problem rather than infrastructure and operations — and all provide integrated security and access control through familiar AWS services: **data is encrypted by default**, protected with the same granular encryption and access controls used throughout the AWS Cloud. Choose based on your use case, your resources, and the skills and experience of the people who will build and manage the application.

TABLE 3.6 Three primary AWS services for generative AI

SERVICE	WHAT IT DOES	STANDOUT CAPABILITIES
Amazon SageMaker AI	Build, train, and deploy ML models — including FMs — in a secure, integrated development environment	Purpose-built tools across the **ML lifecycle** (training infrastructure, governance, ML operations); **SageMaker JumpStart** ML hub deploys publicly available FMs in a few clicks; **full control** over model customization and training/deployment pipelines — the choice for specialized needs and fine-grained control
Amazon Bedrock	Build scalable generative AI applications on popular FMs without building ML infrastructure	Choose, experiment with, and **evaluate** a variety of FMs for your use case; **image and text playgrounds** to test prompts and parameters from the console; every model through a **single API**; your own data for knowledge bases or model customization; **guardrails and content filters** against undesired outputs
Amazon Q	A set of generative AI applications giving end users and non-technical roles genAI capabilities **without building new applications**	**Q Developer**: coding assistant — generate code, explain program logic, identify bug fixes, generate functional tests; **Q Business**: internal chat application combining your **enterprise data** with LLM knowledge; also embedded in AWS services (Amazon Q in QuickSight, Amazon Q in Connect); powered by Bedrock, so it **inherits Bedrock's safety, security, and responsible-AI controls**

ONE-LINE JOBS TO MEMORIZE

- **SageMaker AI** = full ML lifecycle work.
- **Bedrock** = generative AI application development.
- **Amazon Q** = using generative AI without building an application.
- The module includes a recorded demo of **Amazon Q Developer** assisting with software development tasks.

WORKED EXAMPLE – THE MODULE'S SAMPLE EXAM QUESTION

"An AI research team wants to compare the performance of various large language models on a specific natural language understanding task. Which AWS service offers the MOST efficient way to access and evaluate multiple foundation models without managing the underlying infrastructure?" Key phrases: **compare performance**, **access and evaluate multiple FMs**, **without managing infrastructure**. Answer: **Amazon Bedrock** — API access to multiple FMs from different AI companies, allowing easy comparison with no infrastructure to manage. SageMaker AI *could* deploy and manage the models but requires more setup; Amazon Q is an AI assistant for AWS queries and workplace productivity, not a model-comparison platform; Amazon Comprehend is an NLP service that provides no access to multiple FMs.

3.4 Generative AI use cases

A **use case** describes a specific situation in which a product or service could potentially be used. Before choosing generative AI, **work backward from the business problem** and ask first: *is this a good fit for generative AI?* Requirements that suggest yes — and characteristics that suggest no:

GOOD FIT – REQUIREMENTS	POOR FIT – CHARACTERISTICS
generative AI is likely to deliver value	*signals that genAI won't meet the need*
▪ Processing and analyzing **diverse, unstructured data** (text, images, audio) — where genAI beats traditional ML ▪ **Synthesizing complex information** — distilling insights from multiple sources for research, analysis, decision support ▪ Generating **human-like text, creative content, realistic images** ▪ **Repetitive content creation** done efficiently — product descriptions, report generation ▪ **Personalization at scale** — unique outputs per user preference or context	▪ **Ethical concerns** that can't be tolerated or mitigated ▪ **Accuracy and consistency** requirements where creative predictions are dangerous ▪ **Explainability / transparency** expectations — many genAI models are black boxes ▪ **Ill-defined or constantly changing** problems, or data that goes stale fast ▪ Lack of enough **high-quality data** ▪ Lack of necessary **skills or resources** ▪ **Unclear business value** for the cost

The poor-fit list deserves concrete anchors. A public-facing media site for children has near-zero tolerance for offensive content (**ethics**). Incorrect results in medical diagnosis or legal applications can be disqualifying (**accuracy**). Traditional ML models have established explainability methods; many generative models don't, which matters in decisions that affect individuals and in regulated industries (**explainability**). A model generating product descriptions will struggle if the catalog or requirements change frequently (**changing problem**), and generating technical documentation for a brand-new product fails for lack of historical data (**data quality**). Generative AI is new and evolving — even ML-skilled organizations may need extra time and resources, though tools with lower learning curves reduce the barrier (**skills**). And always ask not just whether generative AI *can* do the task, but whether a **less complex or less costly approach** meets the business need (**value for cost**).

Where it fits, generative AI **accelerates innovation** across four business dimensions: improving the **customer experience**, boosting **employee productivity**, enhancing **research and creativity**, and optimizing **operational processes**. A good starting strategy: pick a use case where **metrics are already tracked** and impact is measurable in a short time frame — a reduction in help-desk call times, or an increase in deployments per period. Virtual assistants and chat-based apps cut across several dimensions at once, making both customer-facing and employee-facing systems more efficient; summarization of reports, research notes, or meeting minutes adds efficiency to redundant processes without much overhead.

TABLE 3.7 Generative AI in different roles

ROLE	EXAMPLE USES	HOW IT HELPS THE ROLE
Customer service rep	Generate personalized responses; troubleshoot common issues; provide recommendations; summarize call activity	Respond to more inquiries; more consistent, tailored responses; time freed for complex requests; higher customer-satisfaction ratings
Developer	Generate code snippets; debug code; write unit tests; generate documentation; automate repetitive tasks	Ramp up on new technologies faster; less time hunting samples and troubleshooting; time freed for creative development
Marketing analyst	Generate copy for social media and websites; personalize campaigns; extract key elements from customer data for analysis	Less time producing similar content for different media; consistent tone and messaging; time freed for creative marketing

Amazon itself supplies visible examples. The **Rufus** shopping assistant and **customer review highlights** can be seen directly on Amazon.com. **Amazon Pharmacy** uses generative AI to streamline prescription filling — automatically interpreting handwritten or digital prescriptions, extracting key information such as dosage and frequency, and giving pharmacists clear instructions — and to power customer-service chat apps that answer questions quickly and accurately.

HOW THIS COURSE'S OWN TEAM USED GENERATIVE AI – AND WHAT THEY LEARNED

- Used for: content creation · critiques and suggestions on design documents · examples and distinguishing characteristics of technical concepts · knowledge-check topics · style-guide alignment · captions.
- Payoff: faster initial drafts, shorter review cycles for SMEs and editors, more time for creative work, reduced production time for release and maintenance.
- Take time to **brainstorm and experiment**; practice prompting and **catalog the prompts that work well**.
- **Review all outputs thoroughly** — genAI output alone would not have met requirements.
- Capabilities and tools are a **moving target**; use generative AI as a **support tool, not a replacement for critical thinking**.

COMMON PITFALL

"It can, therefore it should" is the classic use-case mistake. Work backward from the business problem, and reject the genAI route when a simpler, cheaper approach meets the need or when ROI can't be quantified.

3.5 PartyRock, an Amazon Bedrock playground

PartyRock is an Amazon Bedrock playground for building generative AI skills through experimentation. You can **build and share generative AI applications without coding**, and **access and compare FMs without an AWS account** — sign in with an Amazon, Google, or Apple ID and use it with **free daily usage** limits. Search the application catalog to reuse existing applications, or use the **remix** option to customize an existing application for your own use case. The module's activity walks through creating a PartyRock account and running a first prompt that generates both a **text** response and an **image** response from the course's activity app.

COMMUNITY GUIDELINES AND LIMITS

- PartyRock is for **educational and entertainment purposes** — review the community guidelines before use.
- Don't create content that provides (or could be seen as) **medical, health, financial, or legal advice**.
- Don't include **private information** — yours or anyone else's: no phone numbers, email addresses, or mailing addresses.
- It is a playground for trying features and small experimental apps — **not designed for scalable production applications**.

Module summary

- Foundation models are deep learning models on complex neural networks, extensively pre-trained on massive data; their hallmark is **generalized responses** to natural-language input across language, image, audio, and biomedical categories — plus multimodal combinations.

- A **prompt** is a natural-language instruction or question; it plays the role of new data for inference, so every generated output is a **prediction** whose quality depends on training data, the model, and your phrasing.
- **LLMs** are FMs for text (trillions of words) handling text generation, Q&A, summarization, text extraction, conversation with retained context, and code generation.
- Architectures: **diffusion** (add noise, learn to reverse), **GANs** (generator vs. discriminator), **VAEs** (compressed latent representations), **transformers** (parallel sequence processing with attention — the basis of modern LLMs).
- Text becomes **tokens**; tokens become **embeddings** — dense vectors of hundreds to thousands of dimensions where similar meanings sit close — and the model predicts what probably comes next.
- Manage the four output risks — **hallucinations, toxicity, bias, illegality** — with reviews, guardrails, fairness strategies, and audits; tune **inference parameters** (deterministic ↔ creative); escalate prompt → knowledge base → additional training before ever building a model.
- AWS services: **SageMaker AI** = full ML lifecycle (JumpStart hub, pipeline control); **Bedrock** = build genAI apps, many FMs behind one API with playgrounds and guardrails; **Amazon Q** = genAI without building an app (Q Developer for code, Q Business for enterprise chat). All managed, encrypted by default.
- **Work backward from the business problem**; favor measurable use cases across customer experience, productivity, research/creativity, and operations; experiment freely — **PartyRock** offers a no-code, no-account Bedrock playground.

Review questions

1. **What distinguishes a foundation model from other deep learning models, and why does that matter?**

   Answer: The degree to which it produces generalized responses. Trained extensively on massive data, an FM can answer natural-language requests it was never specifically designed for — which is what makes one model powerful across a broad array of applications.

2. **Trace what happens to the text you type into an LLM before it can generate a reply.**

   Answer: Tokenization breaks the text into tokens (words, subwords, or characters, depending on strategy); embedding converts each token into a dense numeric vector with hundreds to thousands of dimensions, where similar tokens sit near one another; the network then uses learned patterns across words, sequences, and context to predict what probably comes next.

3. **Match each architecture to its signature: diffusion, GAN, VAE, transformer.**

   Answer: Diffusion — adds noise to clean input and learns to reverse the process (high-quality images). GAN — a generator creates samples to fool a discriminator that classifies real vs. fake. VAE — learns a compressed latent representation and generates samples resembling the original. Transformer — processes sequences in parallel with simultaneous attention; the architecture behind modern LLMs.

4. **Your HR chatbot keeps inventing policy details. Name the levers to fix it, cheapest first.**

   Answer: Make inference parameters more deterministic and enrich the prompt (instructions, examples, reference material); connect an external knowledge base — the actual internal policy documents — so the model looks answers up; if still short, run additional training on a task-specific dataset. Wrap it all in safeguards: guardrails, human review, auditing.

5. Why are hallucinations tolerable in some applications and disqualifying in others?

Answer: Because outputs are probability-based predictions, they can sound plausible while being verifiably untrue (the classic example: nonexistent scientific citations). Creative storytelling tolerates that; legal briefings — and medical or legal decisions generally — cannot.

6. A team wants internal chat over company documents but can't build or manage applications. Which AWS service fits, and when would the other two be right instead?

Answer: Amazon Q Business — an internal chat application over enterprise data plus LLM knowledge, with no app to build. Choose Bedrock to build a custom genAI application without ML infrastructure (single API, knowledge bases, guardrails); choose SageMaker AI for full ML-lifecycle control — building, training, and deploying your own models.

7. List four signals that generative AI is a poor fit for a business problem.

Answer: Any four of: ethical risks that can't be tolerated or mitigated; strict accuracy/consistency requirements; explainability or transparency expectations (black-box models); an ill-defined or constantly changing problem; lack of high-quality data; lack of skills or resources; unclear business value for the cost.

8. What did the course-development team learn from using generative AI to build this course?

Answer: Brainstorm and experiment; practice prompting and catalog the prompts that work; review all outputs thoroughly, because generative AI output alone didn't meet requirements; treat capabilities and tools as a moving target; and use generative AI as a support tool, never a replacement for expertise and critical thinking.