Generative AI refers to AI capabilities that create original content without explicit instructions. It includes:

* **Natural Language Generation:** Creating text-based responses to user queries or prompts, like writing a cover letter based on specific criteria.
* **Image Generation:** Producing images based on descriptions or requests, such as generating a logo for a business.
* **Code Generation:** Assisting developers by generating code snippets or scripts to perform specific tasks, like writing Python code to add numbers together.

Language models are specialized machine learning models used for natural language processing (NLP) tasks.

 **Transformer Models**: The latest language models, like GPT-4 and BERT, are based on transformer architecture. These models are trained on vast amounts of text data, enabling them to understand and generate human-like language responses.

 **Tokenization**: Training begins with breaking down text into tokens, assigning each token a unique ID. Tokens can represent individual words, parts of words, or combinations of words and punctuation.

 **Embeddings**: Each token is represented by a vector (a multi-valued numeric array) called an embedding. These embeddings capture semantic attributes of the tokens, allowing the model to understand relationships between words.

 **Attention**: Attention layers in the model examine sequences of tokens and determine the strength of relationships between them. Self-attention, a key aspect, considers how tokens influence each other's meaning within a sequence.

 **Training**: During training, the model predicts the next token in a sequence based on the tokens that precede it. This prediction is refined iteratively, with adjustments made to minimize errors and improve accuracy.

 **Inference**: During inference, the trained model generates responses to user prompts by predicting the most probable tokens to follow the input sequence. The generated output appears syntactically correct and often indistinguishable from human-generated text.

 They can determine sentiment, summarize text, compare text sources for similarity, and generate new natural language.

 The latest language models are based on transformer architecture, consisting of encoder and decoder blocks.

 Training begins with tokenization, breaking down text into tokens and assigning unique IDs.

 Embeddings are created for tokens, representing semantic attributes and relationships between words.

 Attention mechanisms in the model examine token sequences, quantifying relationships between them.

 During training, the model predicts the next token in a sequence based on preceding tokens, iteratively refining predictions.

 Inference involves generating responses to user prompts by predicting probable tokens, producing coherent and contextually relevant text.

 Organizations and developers often use existing foundation models for language tasks, rather than training from scratch.

 Azure provides foundation models through the Azure OpenAI service and the Model Catalog, offering secure, scalable hosting.

 The Model Catalog includes models from OpenAI, HuggingFace, and others, such as the GPT collection and DALL-E for image generation.

 Language models can be categorized into Large Language Models (LLMs) and Small Language Models (SLMs).

 LLMs are trained with vast amounts of diverse text and have billions or trillions of parameters, offering comprehensive language generation capabilities.

 SLMs are trained with smaller, focused datasets, have fewer parameters, and are more suitable for specific conversational topics.

 LLMs can be challenging to deploy locally due to their size, while SLMs offer more deployment options, including local and on-premises deployment.

 Copilots are generative AI assistants integrated into applications, often as chat interfaces, to provide contextualized support for common tasks.

 Microsoft Copilot is integrated into various Microsoft applications and user experiences, based on an open architecture allowing third-party developers to create plug-ins and custom copilots.

 Business users utilize copilots to enhance productivity and creativity with AI-generated content and task automation.

 Developers extend copilots by creating plug-ins to integrate them into business processes, data, or even develop custom copilots for specific applications or services.

 Copilots can revolutionize work by assisting with tasks like first drafts, information synthesis, strategic planning, etc., aiming to empower users to be smarter, more productive, creative, and connected.

 Copilot adoption can be categorized into off-the-shelf use (e.g., Microsoft Copilot), extending Microsoft Copilot for custom business processes, or building custom copilots for unique experiences.

 **Web Browsing with AI**: Microsoft Copilot assists in answering questions, creating content, and searching the web, available on the Copilot website, Bing search engine, and Edge browser.

 **AI Assistance for Information Workers**:

* In Microsoft Word, Copilot generates documents based on natural language prompts and refines them.
* In Microsoft PowerPoint, it creates presentations from documents or emails and enhances them.
* In Microsoft Outlook, it summarizes emails, checks schedules, and finds relevant emails/documents for meetings.

 **Use AI to Support Business Processes**:

* **Copilot in Dynamics 365 Customer Service**: Assists in analyzing support tickets, finding resolutions, and communicating solutions.
* **Copilot for Dynamics 365 Sales**: Helps in finding relevant customer and industry information to streamline sales processes.
* **Copilot for Dynamics 365 Supply Chain**: Optimizes procurement decisions by assessing the impact and risk of changes to purchase orders.

 **AI-Assisted Data Analytics**:

* **Copilot in Microsoft Fabric**: Automatically generates code for data analysis, manipulation, and visualization in Spark notebooks.
* **Copilot in Power BI**: Analyzes data and suggests appropriate data visualizations for Power BI reports.

 **Manage IT Infrastructure and Security**:

* **Microsoft Copilot for Security**: Assists security professionals in assessing, mitigating, and responding to security threats.
* **Microsoft Copilot for Azure**: Integrated into the Azure portal to support infrastructure administrators in working with Azure cloud services.

 **AI-Powered Software Development**:

* **GitHub Copilot**: Enhances developer productivity by analyzing and explaining code, adding documentation, and generating new code based on natural language prompts.

 To optimize Copilot responses:

* Start with a specific goal.
* Provide a source for grounded information.
* Add context to maximize relevance.
* Set clear expectations for format, tone, and style.
* Iterate based on previous responses for refinement.

 Copilot augments prompts with system messages, conversation history, and potentially optimized wording.

 Prompt engineering is the process of refining prompts to enhance generative AI response quality.

* **Copilot Studio**:
  + Designed for low-code development scenarios.
  + Allows technically proficient users to create conversational AI experiences.
  + Hosted in Microsoft 365 environment, accessible through chat channels like Teams.
  + Fully managed SaaS solution, simplifying infrastructure and deployment concerns.
  + Focus on creating effective solutions without worrying about deployment details.
  + More information: [Copilot Studio](https://www.microsoft.com/microsoft-copilot/microsoft-copilot-studio).
* **Azure AI Studio**:
  + A PasS development portal for professional software developers.
  + Provides full control over the language model, including fine-tuning with custom data.
  + Allows defining prompt flows, integrating data augmentation, and prompt engineering logic.
  + Enables deployment of copilot service in the cloud, consumable from custom apps and services.
  + More information: [Azure AI Studio](https://azure.microsoft.com/products/ai-studio/).

To plan a responsible generative AI solution, follow these four stages:

1. **Identify potential harms**: Understand the possible negative consequences that could arise from your solution, considering ethical, legal, and societal implications.
2. **Measure harm presence**: Assess the extent to which these harms manifest in the outputs generated by your solution through thorough monitoring and evaluation.
3. **Mitigate harms**: Implement measures at various layers of your solution to minimize the occurrence and impact of identified harms. Ensure transparency in communicating risks to users.
4. **Operate responsibly**: Define and adhere to a deployment and operational readiness plan that outlines protocols for managing risks and maintaining accountability throughout the solution's lifecycle.

By following these stages, you can develop and implement a responsible generative AI solution that aligns with ethical standards and regulatory requirements while mitigating potential risks to users and society.

* **Identify potential harms**:
  + Consider offensive or discriminatory content, factual inaccuracies, and unethical behavior.
  + Consult documentation from service providers and model developers.
* **Prioritize identified harms**:
  + Assess likelihood and impact, prioritizing based on potential impact and likelihood considering both intended and potential misuse.
* **Test and verify prioritized harms**:
  + Conduct testing to verify the presence of identified harms.
  + Use methods like "red team" testing to probe for weaknesses and harmful outcomes.
* **Document and share verified harms**:
  + Document evidence supporting potential harms.
  + Share details with stakeholders and maintain a prioritized list of identified harms, updating it as needed.

 **Prepare input prompts**:

* Create diverse prompts likely to result in each potential harm documented for the system.
* For example, if a potential harm is generating dangerous poisons, create prompts asking how to create such substances.

 **Generate output**:

* Submit the prepared prompts to the system and retrieve the generated output.

 **Evaluate output**:

* Apply predefined criteria to categorize the output based on the level of potential harm it contains.
* Criteria can range from simple "harmful" or "not harmful" labels to more nuanced harm levels.

 **Document and share results**:

* Document the results of the measurement process.
* Share findings with stakeholders.

 **Manual and automatic testing**:

* Start with manual testing to ensure consistency and define evaluation criteria.
* Automate testing for larger volumes of test cases, potentially using a classification model.
* Periodically perform manual testing to validate new scenarios and ensure the automated solution's effectiveness.

Here's how you can mitigate potential harms in a generative AI solution across different layers:

* **Model Layer**:
  + Choose a model appropriate for the solution's use case.
  + Fine-tune foundational models with specific training data to make responses more relevant and scoped.
* **Safety System Layer**:
  + Utilize platform-level configurations like content filters to suppress harmful prompts and responses.
  + Implement abuse detection algorithms to identify systematic abuse.
  + Enable alert notifications for quick response to potential harmful behavior.
* **Metaprompt and Grounding Layer**:
  + Define metaprompts or system inputs to set behavioral parameters for the model.
  + Apply prompt engineering to add grounding data, enhancing relevance and reducing harm.
  + Use retrieval augmented generation (RAG) to include contextual data from trusted sources in prompts.
* **User Experience Layer**:
  + Design the application UI to constrain inputs and validate both inputs and outputs.
  + Ensure documentation transparently communicates the system's capabilities, limitations, and potential harms.

By implementing mitigations at each layer, you can reduce the risk of harmful outputs and enhance the safety and reliability of your generative AI solution.

Before releasing and operating a responsible generative AI solution, consider the following steps:

1. **Complete Prerelease Reviews**:
   * Conduct compliance reviews covering legal, privacy, security, and accessibility aspects.
   * Ensure relevant teams review the system and its documentation to meet compliance requirements.
2. **Release and Operate the Solution**:
   * Devise a phased delivery plan, starting with a restricted group of users for initial feedback and issue identification.
   * Develop an incident response plan with estimated response times for unforeseen incidents.
   * Establish a rollback plan outlining steps to revert the solution in case of issues.
   * Implement mechanisms to immediately block harmful responses and users in case of misuse.
   * Enable users to provide feedback and report issues, categorizing content as inaccurate, incomplete, harmful, or offensive.
   * Track telemetry data to assess user satisfaction, identify functional gaps, and address usability challenges, ensuring compliance with privacy laws and organizational policies.

By following these guidelines, you can ensure a successful release and ongoing operation of your generative AI solution while prioritizing user safety and satisfaction.