```
!pip install google-generativeai
```

```
Requirement already satisfied: google-generativeai in
/usr/local/lib/python3.10/dist-packages (0.3.2)
Requirement already satisfied: google-ai-generativelanguage==0.4.0
in /usr/local/lib/python3.10/dist-packages (from google-generativeai)
(0.4.0)
Requirement already satisfied: google-auth in
/usr/local/lib/python3.10/dist-packages (from google-generativeai)
(2.27.0)
Requirement already satisfied: google-api-core in
/usr/local/lib/python3.10/dist-packages (from google-generativeai)
(2.11.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from google-generativeai)
(4.11.0)
Requirement already satisfied: protobuf in
/usr/local/lib/python3.10/dist-packages (from google-generativeai)
(3.20.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from google-generativeai) (4.66.2)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in
/usr/local/lib/python3.10/dist-packages (from google-ai-
generativelanguage==0.4.0->google-generativeai) (1.23.0)
Requirement already satisfied: googleapis-common-
protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3.10/dist-packages
(from google-api-core->google-generativeai) (1.63.0)
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in
/usr/local/lib/python3.10/dist-packages (from google-api-core->google-
generativeai) (2.31.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth->google-
generativeai) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth->google-
generativeai) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth->google-
generativeai) (4.9)
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in
/usr/local/lib/python3.10/dist-packages (from google-api-core->google-
generativeai) (1.62.1)
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in
/usr/local/lib/python3.10/dist-packages (from google-api-core->google-
generativeai) (1.48.2)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth->google-generativeai) (0.6.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-core->google-generativeai)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-core->google-generativeai)
(3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-core->google-generativeai)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3.0.0.dev0,>=2.18.0->google-api-core->google-generativeai)
(2024.2.2)
```

```
import google.generativeai as genai
import json
import pathlib
import pprint
import  requests
from IPython.display import Markdown
```

```
from google.colab import userdata
API_KEY=userdata.get('Silviya')
genai.configure(api_key=API_KEY)
```

```
import google.generativeai as genai
generation_config={
    "temperature":0.9,
    "top_p":1,
    "top_k":1,
    "max_output_tokens":2048,
}
model=genai.GenerativeModel(model_name="gemini-
pro",generation_config=generation_config,
                            )
prompt_parts=[
    input("Enter Your Prompt: "),
]
response=model.generate_content(prompt_parts,stream=False)
```

```
Enter Your Prompt: india
```

```
Markdown(response.text)
```

```
<IPython.core.display.Markdown object>
```

```
from IPython.display import Markdown
```

```
pip install pypdf
```

```
Collecting pypdf
  Downloading pypdf-4.2.0-py3-none-any.whl (290 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/290.4 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━ ━━━━━━━━━━━━━━━━━━━━ 143.4/290.4 kB 4.2 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 290.4/290.4 kB 5.1
MB/s eta 0:00:00
ent already satisfied: typing_extensions>=4.0 in
/usr/local/lib/python3.10/dist-packages (from pypdf) (4.11.0)
Installing collected packages: pypdf
Successfully installed pypdf-4.2.0
```

```
from pypdf import PdfReader
pdf = PdfReader('AWS developer guide.pdf')
text = ''
for page in pdf.pages:
  text += page.extract_text()
Markdown(text)
```

```
<IPython.core.display.Markdown object>
```

```
import google.generativeai as genai
generation_config={
    "temperature":0.9,
    "top_p":1,
    "top_k":1,
    "max_output_tokens":2048,
}
model=genai.GenerativeModel(model_name="gemini-
pro",generation_config=generation_config,
                            )
```

```
prompt=input("Enter Your Prompt:")
responce=chat.send_message(f"{prompt},As per the above document")
Markdown(responce.text)
```

```
Enter Your Prompt:aws developer guide
```

```
<IPython.core.display.Markdown object>
```

```
responce=chat.send_message(input("Enter Your Promt:"))
Markdown(responce.text)
```

```
Enter Your Promt:topics
```

```
<IPython.core.display.Markdown object>
```

```
chat=model.start_chat(history=[])
responce=chat.send_message(text)
```

```
ERROR:tornado.access:500 POST /v1beta/models/gemini-
pro:generateContent?%24alt=json%3Benum-encoding%3Dint (127.0.0.1)
8712.14ms
```

```
---------------------------------------------------------------------------
InternalServerError                       Traceback (most recent call
last)
<ipython-input-18-0eb2426e82fd> in <cell line: 2>()
      1 chat=model.start_chat(history=[])
----> 2 responce=chat.send_message(text)

/usr/local/lib/python3.10/dist-packages/google/generativeai/generative
_models.py in send_message(self, content, generation_config,
safety_settings, stream, **kwargs)
    365         if generation_config.get("candidate_count", 1) > 1:
    366             raise ValueError("Can't chat with `candidate_count
> 1`")
--> 367         response = self.model.generate_content(
    368             contents=history,
    369             generation_config=generation_config,

/usr/local/lib/python3.10/dist-packages/google/generativeai/generative
_models.py in generate_content(self, contents, generation_config,
safety_settings, stream, **kwargs)
    246             return
generation_types.GenerateContentResponse.from_iterator(iterator)
    247         else:
--> 248             response = self._client.generate_content(request)
    249             return
generation_types.GenerateContentResponse.from_response(response)
    250

/usr/local/lib/python3.10/dist-packages/google/ai/generativelanguage_v
1beta/services/generative_service/client.py in generate_content(self,
request, model, contents, retry, timeout, metadata)
    564
    565         # Send the request.
--> 566         response = rpc(
    567             request,
    568             retry=retry,

/usr/local/lib/python3.10/dist-packages/google/api_core/gapic_v1/metho
d.py in __call__(self, timeout, retry, *args, **kwargs)
    111             kwargs["metadata"] = metadata
    112
--> 113         return wrapped_func(*args, **kwargs)
    114
    115

/usr/local/lib/python3.10/dist-packages/google/api_core/retry.py in
retry_wrapped_func(*args, **kwargs)
    347                 self._initial, self._maximum,
multiplier=self._multiplier
```

```
    348             )
--> 349             return retry_target(
    350                 target,
    351                 self._predicate,

/usr/local/lib/python3.10/dist-packages/google/api_core/retry.py in
retry_target(target, predicate, sleep_generator, timeout, on_error,
**kwargs)
    189     for sleep in sleep_generator:
    190         try:
--> 191             return target()
    192 
    193         # pylint: disable=broad-except

/usr/local/lib/python3.10/dist-packages/google/api_core/timeout.py in
func_with_timeout(*args, **kwargs)
    118                 kwargs["timeout"] = max(0, self._timeout -
time_since_first_attempt)
    119 
--> 120             return func(*args, **kwargs)
    121 
    122         return func_with_timeout

/usr/local/lib/python3.10/dist-packages/google/api_core/grpc_helpers.p
y in error_remapped_callable(*args, **kwargs)
     70     def error_remapped_callable(*args, **kwargs):
     71         try:
---> 72             return callable_(*args, **kwargs)
     73         except grpc.RpcError as exc:
     74             raise exceptions.from_grpc_error(exc) from exc

/usr/local/lib/python3.10/dist-packages/google/ai/generativelanguage_v
1beta/services/generative_service/transports/rest.py in __call__(self,
request, retry, timeout, metadata)
    854             # subclass.
    855             if response.status_code >= 400:
--> 856                 raise
core_exceptions.from_http_response(response)
    857 
    858             # Return the response

InternalServerError: 500 POST
https://generativelanguage.googleapis.com/v1beta/models/gemini-
pro:generateContent?%24alt=json%3Benum-encoding%3Dint: An internal
error has occurred. Please retry or report in
https://developers.generativeai.google/guide/troubleshooting

chat.history
```

```
[parts {
   text: "aws"
 }
 role: "user",
 parts {
   text: "**What is AWS (Amazon Web Services)?**\n\nAmazon Web
Services (AWS) is a comprehensive, on-demand cloud computing platform
that provides scalable infrastructure and a suite of cloud services to
businesses, governments, and individuals.\n\n**Key Features and
Services:**\n\n* **Compute:** Elastic Compute Cloud (EC2), Amazon EC2
Container Service\n* **Storage:** Amazon Simple Storage Service (S3),
Amazon Elastic Block Store (EBS)\n* **Networking:** Amazon Virtual
Private Cloud (VPC), Amazon Route 53\n* **Security:** Amazon
GuardDuty, AWS Shield\n* **Databases:** Amazon Relational Database
Service (RDS), Amazon DynamoDB\n* **Analytics:** Amazon Redshift,
Amazon Athena\n* **Machine Learning:** Amazon SageMaker, Amazon
Rekognition\n* **Internet of Things (IoT):** AWS IoT Core, AWS
Greengrass\n* **Artificial Intelligence:** Amazon Lex, Amazon Polly\n\
n**Benefits of AWS:**\n\n* **Scalability:** Elastic infrastructure
that scales automatically based on demand.\n* **Cost Savings:** Pay-
as-you-go model eliminates upfront capital expenses.\n*
**Reliability:** Globally distributed data centers with high
availability and fault tolerance.\n* **Security:** Comprehensive
security features to protect data and applications.\n* **Innovation:**
Continuous updates and new service releases foster innovation.\n*
**Flexibility:** Supports a wide range of operating systems,
programming languages, and application frameworks.\n\n**Use Cases:**\
n\nAWS is used by organizations of all sizes and industries for
various purposes, including:\n\n* **Web and mobile app hosting**\n*
**Data storage and backup**\n* **Enterprise applications**\n*
**Machine learning and AI**\n* **IoT device management**\n* **Big data
analytics**\n* **Cloud-based gaming**\n\n**Pricing:**\n\nAWS offers a
pay-as-you-go pricing model, allowing customers to pay only for the
services and resources they use. Pricing varies depending on the
service and region.\n\n**Conclusion:**\n\nAWS provides a powerful
suite of cloud computing services that empower businesses and
individuals to innovate, scale, and secure their applications. Its
cost-effectiveness, reliability, and scalability make it a leading
choice for organizations looking to leverage the benefits of cloud
computing."
 }
 role: "model",
 parts {
   text: "topics,As per the above document"
 }
 role: "user",
 parts {
   text: "**Key Topics Covered in the Document on AWS:**\n\n*
Definition and overview of AWS (Amazon Web Services)\n* Key features
and services offered by AWS\n* Benefits of using AWS\n* Common use
```

```
cases for AWS\n* AWS pricing model\n\n**Additional Topics:**\n\n*
**Cloud Computing Concepts:** The document assumes a basic
understanding of cloud computing concepts such as scalability,
elasticity, and virtualization.\n* **Security and Compliance:** AWS
offers a wide range of security features and compliance
certifications, which are briefly mentioned but not covered in
detail.\n* **AWS Regions and Availability Zones:** AWS has a global
network of data centers spread across multiple regions and
availability zones, which are not discussed in the document.\n* **AWS
Management Console and Tools:** AWS provides a user-friendly
management console and various tools for managing and monitoring cloud
resources, which are not mentioned in the document.\n* **AWS
Marketplace:** AWS Marketplace offers a wide variety of software,
services, and data products from third-party vendors, which is not
covered in the document.\n\nThe document provides a concise overview
of AWS and its key offerings. For more detailed information on
specific topics, it is recommended to refer to the official AWS
documentation or consult with an AWS expert."
 }
 role: "model",
 parts {
   text: "topics"
 }
 role: "user",
 parts {
   text: "**Key Topics Covered in the Document on AWS:**\n\n*
Definition and overview of AWS (Amazon Web Services)\n* Key features
and services offered by AWS\n* Benefits of using AWS\n* Common use
cases for AWS\n* AWS pricing model\n\n**Additional Topics:**\n\n*
**Cloud Computing Concepts:** The document assumes a basic
understanding of cloud computing concepts such as scalability,
elasticity, and virtualization.\n* **Security and Compliance:** AWS
offers a wide range of security features and compliance
certifications, which are briefly mentioned but not covered in
detail.\n* **AWS Regions and Availability Zones:** AWS has a global
network of data centers spread across multiple regions and
availability zones, which are not discussed in the document.\n* **AWS
Management Console and Tools:** AWS provides a user-friendly
management console and various tools for managing and monitoring cloud
resources, which are not mentioned in the document.\n* **AWS
Marketplace:** AWS Marketplace offers a wide variety of software,
services, and data products from third-party vendors, which is not
covered in the document.\n\n**Summary of Key Topics:**\n\n**1.
Definition and Overview of AWS:**\n   - What is AWS and what services
does it provide?\n\n**2. Key Features and Services:**\n   - Compute,
storage, networking, security, databases, analytics, machine learning,
IoT, AI\n\n**3. Benefits of Using AWS:**\n   - Scalability, cost
savings, reliability, security, innovation, flexibility\n\n**4. Common
Use Cases:**\n   - Web and mobile app hosting, data storage and
backup, enterprise applications, machine learning and AI, IoT device
```

```
management, big data analytics, cloud-based gaming\n\n**5. AWS Pricing
Model:**\n   - Pay-as-you-go model, pricing varies depending on
service and region\n\n**Additional Considerations:**\n\n*
Understanding cloud computing concepts is helpful for fully
comprehending AWS.\n* AWS offers robust security features and
compliance certifications, which are important for protecting data and
applications.\n* AWS has a global infrastructure with multiple regions
and availability zones, ensuring high availability and reliability.\n*
AWS provides a user-friendly management console and tools for managing
and monitoring cloud resources.\n* AWS Marketplace offers a wide range
of third-party products and services, extending the capabilities of
AWS.\n\nBy understanding these key topics, you can gain a solid
foundation in AWS and its offerings."
 }
 role: "model",
 parts {
   text: "aws,As per the above document"
 }
 role: "user",
 parts {
   text: "**Key Concepts and Terminology Related to AWS:**\n\n*
**Cloud Computing:** AWS is a cloud computing platform that provides
on-demand access to computing resources over the internet.\n*
**Scalability:** AWS allows you to scale your applications and
infrastructure up or down as needed, without the need for upfront
investment in hardware.\n* **Elasticity:** AWS resources can be
provisioned and released automatically based on demand, ensuring that
you only pay for the resources you use.\n* **Virtualization:** AWS
uses virtualization technology to create isolated and secure
environments for running applications.\n* **Regions and Availability
Zones:** AWS has a global network of data centers located in multiple
regions and availability zones to ensure high availability and
reliability.\n* **Services:** AWS offers a wide range of services,
including compute, storage, networking, security, databases,
analytics, machine learning, IoT, and AI.\n* **Pay-as-you-go
Pricing:** AWS follows a pay-as-you-go pricing model, where you only
pay for the services and resources you use.\n\n**Key Benefits of Using
AWS:**\n\n* **Cost Savings:** AWS eliminates the need for upfront
capital expenses and allows you to pay only for the resources you
use.\n* **Scalability and Elasticity:** AWS provides the ability to
scale your applications and infrastructure up or down as needed,
without any downtime.\n* **Reliability and Availability:** AWS has a
global network of data centers and multiple availability zones to
ensure high availability and reliability for your applications.\n*
**Security:** AWS offers a wide range of security features and
compliance certifications to protect your data and applications.\n*
**Innovation:** AWS is constantly innovating and releasing new
services and features, allowing you to stay at the forefront of
technology.\n* **Flexibility:** AWS supports a wide range of operating
systems, programming languages, and application frameworks, giving you
```

```
the flexibility to build and deploy applications using the tools you
prefer.\n\n**Common Use Cases for AWS:**\n\n* Web and mobile app
hosting\n* Data storage and backup\n* Enterprise applications\n*
Machine learning and AI\n* IoT device management\n* Big data
analytics\n* Cloud-based gaming\n\n**Conclusion:**\n\nAWS is a
comprehensive cloud computing platform that provides a wide range of
services and benefits to businesses and individuals. By understanding
the key concepts and terminology related to AWS, you can leverage its
capabilities to innovate, scale, and secure your applications and
infrastructure."
 }
 role: "model",
 parts {
   text: "aes devoper guide,As per the above document"
 }
 role: "user",
 parts {
   text: "**Key Concepts and Terminology Related to AES Developer
Guide:**\n\n* **AES (Advanced Encryption Standard):** A symmetric
block cipher that is widely used for encrypting and decrypting data.\
n* **Block Cipher:** A type of encryption algorithm that operates on
fixed-size blocks of data.\n* **Key:** A secret value used to encrypt
and decrypt data.\n* **Initialization Vector (IV):** A random value
used to initialize the encryption process and prevent identical
plaintext from producing identical ciphertext.\n* **Ciphertext:** The
encrypted form of plaintext.\n* **Plaintext:** The original,
unencrypted data.\n* **Encryption:** The process of converting
plaintext into ciphertext.\n* **Decryption:** The process of
converting ciphertext back into plaintext.\n* **Key Schedule:** A set
of round keys derived from the original key, used in the encryption
and decryption process.\n* **Round:** A single iteration of the
encryption or decryption algorithm.\n* **S-Box:** A substitution box
used in the encryption and decryption process to provide non-
linearity.\n\n**Key Steps in AES Encryption and Decryption:**\n\
n**Encryption:**\n\n1. Convert the plaintext into a block of 128
bits.\n2. Apply the initial round key to the block.\n3. Perform a
series of rounds (typically 10, 12, or 14), each consisting of:\n    *
SubBytes: Substitute each byte in the block using an S-Box.\n    *
ShiftRows: Shift the rows of the block cyclically.\n    * MixColumns:
Perform a linear transformation on the columns of the block.\n    *
AddRoundKey: Apply the round key to the block.\n4. Apply the final
round key to the block.\n\n**Decryption:**\n\n1. Convert the
ciphertext into a block of 128 bits.\n2. Apply the final round key to
the block (in reverse order).\n3. Perform a series of inverse rounds,
each consisting of:\n    * AddRoundKey: Apply the round key to the
block (in reverse order).\n    * MixColumns: Perform an inverse linear
transformation on the columns of the block.\n    * ShiftRows: Shift
the rows of the block cyclically in reverse order.\n    * SubBytes:
Substitute each byte in the block using an inverse S-Box.\n4. Apply
the initial round key to the block (in reverse order).\n\
n**Conclusion:**\n\nThe AES Developer Guide provides detailed
information on the AES encryption and decryption algorithms, including
the mathematical operations involved and implementation
considerations. By understanding the key concepts and steps involved
in AES, developers can effectively use it to protect sensitive data in
their applications and systems."
 }
 role: "model",
 parts {
   text: "topics"
 }
 role: "user",
 parts {
   text: "**Key Topics Covered in the AES Developer Guide:**\n\n*
Overview of AES (Advanced Encryption Standard)\n* AES Encryption
Algorithm\n* AES Decryption Algorithm\n* Key Expansion and Round Keys\
n* Implementation Considerations\n* Security Analysis and Attacks\n\
n**Additional Topics:**\n\n* **Modes of Operation:** AES can be used
in different modes of operation, such as ECB, CBC, and CTR, to provide
additional security and flexibility.\n* **Padding Schemes:** Padding
schemes are used to ensure that the plaintext is a multiple of the
block size before encryption.\n* **Key Management:** Proper key
management practices are essential to ensure the security of AES
encryption.\n* **Performance Optimization:** Techniques for optimizing
the performance of AES encryption and decryption algorithms.\n*
**Hardware Implementations:** Hardware implementations of AES can
provide significant performance improvements for high-throughput
applications.\n\n**Summary of Key Topics:**\n\n**1. Overview of
AES:**\n   - History, design principles, and applications of AES\n\
n**2. AES Encryption Algorithm:**\n   - Step-by-step description of
the encryption process, including SubBytes, ShiftRows, MixColumns, and
AddRoundKey\n\n**3. AES Decryption Algorithm:**\n   - Step-by-step
description of the decryption process, including inverse operations of
SubBytes, ShiftRows, MixColumns, and AddRoundKey\n\n**4. Key Expansion
and Round Keys:**\n   - Derivation of round keys from the original key
using the key expansion algorithm\n\n**5. Implementation
Considerations:**\n   - Data representation, endianness, and memory
alignment for efficient AES implementation\n\n**6. Security Analysis
and Attacks:**\n   - Known attacks against AES and countermeasures to
mitigate them\n\n**Additional Considerations:**\n\n* Understanding
modes of operation, padding schemes, and key management is important
for secure and efficient use of AES.\n* Performance optimization
techniques can improve the throughput of AES implementations.\n*
Hardware implementations can provide significant performance
advantages for high-volume encryption tasks.\n\nBy understanding these
key topics, developers can effectively implement AES encryption and
decryption in their applications and systems to protect sensitive
data."
 }
 role: "model",
 parts {
   text: "aws developer guide,As per the above document"
 }
 role: "user",
 parts {
   text: "**Key Concepts and Terminology Related to AWS Developer
Guide:**\n\n* **AWS (Amazon Web Services):** A comprehensive cloud
computing platform that provides a wide range of services to
businesses and individuals.\n* **Developer Guide:** A technical
documentation that provides detailed instructions and best practices
for developing applications on a specific platform or using a specific
technology.\n* **SDK (Software Development Kit):** A set of tools and
libraries that allows developers to interact with AWS services
programmatically.\n* **API (Application Programming Interface):** A
set of protocols and routines that define how applications interact
with each other or with external services.\n* **Credentials:** Unique
identifiers and secret keys used to authenticate and authorize access
to AWS services.\n* **Regions and Availability Zones:** AWS has a
global network of data centers located in multiple regions and
availability zones to ensure high availability and reliability.\n*
**Services:** AWS offers a wide range of services, including compute,
storage, networking, security, databases, analytics, machine learning,
IoT, and AI.\n\n**Key Steps for Developing on AWS:**\n\n1. **Create an
AWS account:** Sign up for an AWS account and set up your
credentials.\n2. **Choose a region:** Select the AWS region where you
want to deploy your application.\n3. **Set up an IAM role:** Create an
IAM role to grant your application the necessary permissions to access
AWS services.\n4. **Install the AWS SDK:** Install the appropriate AWS
SDK for your programming language.\n5. **Develop and test your
application:** Write your application code and test it locally.\n6.
**Deploy your application to AWS:** Deploy your application to the
chosen AWS region.\n7. **Monitor and manage your application:** Use
AWS tools to monitor and manage your application\'s performance and
health.\n\n**Conclusion:**\n\nThe AWS Developer Guide provides
comprehensive guidance on how to develop and deploy applications on
the AWS platform. By understanding the key concepts and following the
recommended steps, developers can leverage the power of AWS to build
scalable, reliable, and secure applications."
 }
 role: "model"]
```