**🔹 google-generativeai:-**

google-generativeai ek **Python library** hai jo **Google Gemini** LLM se baat karne ka official tariqa deti hai.

|  |
| --- |
| uv pip install google-generativeai |

|  |  |
| --- | --- |
| GenerativeModel | Gemini ka brain — tumhara model |
| configure | Tumhara API key set karta hai |
| model.generate\_content(...) | Gemini ko sawal bhejta hai |

**Humne apni khud ki Agent class nahi banaye,** balki ek **simple helper class (**PDFReaderAgent**)** banayi hai jo:

* PDF read kare
* Gemini se sawal ka jawab le aaye

📌 **LLM PDF file directly nahi samajhta.**

1. **PDF se text nikaalna hota hai** (ek dafa RAM mein)
2. **Us text ka embedding banana hota hai** (vector ban jata hai)
3. **Wo vector store hota hai** (FAISS, ChromaDB, etc.)
4. Jab user kuch poochta hai:  
   → Query embedding banti hai  
   → Milta-julta vector retrieve hota hai  
   → Sirf related context LLM ko diya jata hai

🧠 LLM ko **na poori PDF milti hai**, na direct file.  
**Sirf relevant chunk/text milta hai RAM se via vector search.**

(PDF → text → embedding → vector DB → query → context → LLM),  
ye **OpenAI ka standard RAG (Retrieval-Augmented Generation)**

**🔹 Short & Important Steps:**

✅ **🔹 Step 1: Load API Key:**

|  |
| --- |
| load\_dotenv()  genai.configure(api\_key=os.getenv("GEMINI\_API\_KEY"))Gemini API key set karo |

Ye .env file ko load karta hai jisme sensitive info hoti hai (jaise API key) Jab load\_dotenv() chalega, to .env file ke andar jo key likhi hai wo memory mein chali jati hai — safe way se.

### genai.configure(api\_key=...)

**Library:** google.generativeai  
**Kaam:** Ye line Gemini ko kehti hai:

“Ye lo API key — ab tum ready ho Gemini model run karne ke liye.”

✅ **🔹 Step 2: Load Gemini Model:**

|  |
| --- |
| model = genai.GenerativeModel(model\_name="models/gemini-2.0-flash") |

### 🔸 1. genai :

* Ye **library ka naam** hai → google.generativeai
* Aap ne uv pip install google-generativeaise install ki thi
* Ye Google Gemini ke models ko **Python code se access** karne deti hai

### 🔸 2. .GenerativeModel(...) :

Ye **ek class** hai jo:

* Gemini ka koi specific version (like "gemini-pro" ya "gemini-2.0-flash") **load** karti hai
* Ye class banati hai **ek model object**, jiske through aap generate\_content(...) jaisa method chala sakti ho

✅ **🔹 Step 3: PDFReaderAgent Class:**

### 🔹 1. class PDFReaderAgent :

* **Ye ek custom class** hai — tumne banayi apne kaam ke liye "Ye agent sirf PDF files ko read karega"

📦 Is class ka kaam:

* PDF file se text nikaalna
* Wo text Gemini ko dena
* Aur sawal ka jawab lena

### 🔹 2. def \_\_init\_\_(self, pdf\_path):

🧠 Isay kehte hain **constructor**  
Jab bhi tum new object banati ho:

To ye \_\_init\_\_ function **automatic run** hota hai.

|  |
| --- |
| agent = PDFReaderAgent("Class\_5.pdf") |

### 🔹 3. self.pdf\_path = pdf\_path :

* Tumne jo file ka naam diya (e.g. "Class\_5.pdf"), wo object ke andar store ho gaya.
* Baad mein jab bhi tum self.pdf\_path likhogi, us file ka naam mil jaega.

### 🔹 4. self.pdf\_text = self.\_extract\_pdf\_text() :

|  |
| --- |
| self.\_extract\_pdf\_text() |

|  |
| --- |
| Class ke andar defined ek method/function ko call kiya ja raha hai |

Jab object banega, ye line chalegi, aur:

* 1. PDF file kholegi
  2. Saari text extract karegi
  3. Us text ko self.pdf\_text mein save karegi

🔍 self.pdf\_text ab **pure PDF ka clean text** ban gaya

✅ 📦 Private Method: \_extract\_pdf\_text() :

* fitz.open() ek **function** hai jo fitz module ke andar hota hai.
* Iska kaam hai: **PDF file open karna** taake usko read ya modify kar sako.

🟢 pdf\_path **kya hai?**

|  |
| --- |
| def \_extract\_pdf\_text(self):  text = ""  doc = fitz.open(self.pdf\_path)  for page in doc:  text += page.get\_text()  doc.close()  return text |

### 🔹 extract\_pdf\_text() kya hai?

* Ye ek **helper function** hai
* Naam ke aage \_ lagaya gaya hai, iska matlab:

“Ye sirf class ke andar use hoga — bahar se nahi”

| **Line** | **Kaam** |
| --- | --- |
| text = "" | Ek khaali string banayi — ismein hum PDF ka text jama karenge |
| doc = fitz.open(self.pdf\_path) | fitz (yaani PyMuPDF) se PDF file open ki |
| for page in doc: | Har page loop mein liya |
| text += page.get\_text() | Page ka text extract karke add kiya |
| doc.close() | File close kar di (achi practice) |
| return text | Sab text return kar diya — jata hai self.pdf\_text mein |

**get\_text()** ek **built-in method hai** jo **PDF file ke text ko extract** karne ke liye use hota hai.  
Ye method aata hai **PyMuPDF library** (jo fitz ke naam se import hoti hai) .

### ****Extracted text**** ka matlab hota hai:

Jaise hm apni PDF file mein kuch lines likhti hein  
program un lines ko **automatically** read karega, aur **text ke form mein** nikaal lega.

## ✅ **🔹 Step 4:** ask() **Method (Inside PDFReaderAgent Class):**

|  |
| --- |
| def ask(self, question):  prompt = f"""  You are a smart PDF assistant. Only answer using the information in this PDF:  {self.pdf\_text}  Now answer this question:  {question}  """  response = model.generate\_content(prompt)  return response.text |

* Ye method **user ka sawal lega** (e.g. "chapter 1 kis page pe hai?")
* Pure PDF ka text + sawal combine karega ek **prompt** mein
* Prompt ko **Gemini model ko bhejega**
* Gemini ka jawab **return karega**

### 🔸 Line-by-Line Breakdown:

#### 1. def ask(self, question):

* Ye class ka **public method** hai
* Jab tum likhogi:

|  |
| --- |
| agent.ask("Yeh PDF kis bare mein hai?") |

To ye method chalega

* question: user ka input (jo tum pooch rahi ho PDF se)

#### 2. prompt = f"""..."""

* **Prompt** woh text hota hai jo Gemini ko bheja jata hai
* Yahan pe f"""...""" means **formatted multi-line string** (f-string)

**Prompt ke andar kya hai?**

|  |
| --- |
| You are a smart PDF assistant.  Only answer using the information in this PDF:  << yahan poora PDF ka text ayega >>  Now answer this question:  << user ka sawal yahan ayega >> |

* {self.pdf\_text} → ismein poora PDF ka extracted text ayega
* {question} → user ka sawal

👀 Is prompt ka goal hai:

Gemini ko force karna ke wo **sirf PDF ke andar se hi jawab de** — apni imagination se nahi.

#### 3. response = model.generate\_content(prompt)

* Ye Gemini Flash model ko **prompt bhejta hai**
* Model ka method .generate\_content(...) chal raha hai
* Jo response milta hai wo response variable mein store hota hai

#### 4. return response.text

* Ye line Gemini ka **sirf text part** wapas return karti hai
* Final result tum print ya use kar sakti ho

### ✅ Final Flow:

|  |
| --- |
| Tum likhti ho → agent.ask("Question?") |

⤷ Gemini ko jaata hai PDF + Question

⤷ Gemini ka jawab milta hai

⤷ Tumhe return hota hai: answer

## ✅ **🔹 Step 5: Use the Agent (Main Block)**

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  agent = PDFReaderAgent("Class\_5.pdf")  question = input("📄 PDF se kya poochna hai? Sawal likho: ")  answer = agent.ask(question)  print("\n🤖 Agent ka jawab:")  print(answer) |

Yeh wo part hai jahan se hamari **custom class PDFReaderAgent** ka **object banta hai**, aur us object ki help se hum Gemini se sawaal karte hain.

### 🔹 Line 1: if \_\_name\_\_ == "\_\_main\_\_":

📌 **Iska matlab kya hai?**  
Ye Python ka special block hota hai. Jab file directly run hoti hai (not imported as a module), to sirf tab hi is block ka code chalta hai.

🧠 Tum samajh lo:  
Agar yeh file **direct run karo**, to niche ka code chalega. Agar **import karo** doosri file me, to nahi chalega.

|  |  |
| --- | --- |
| pdf\_reader.py file:  print("👀 Main file start")  def say\_hello():  print("Hello from PDF Reader!")  if \_\_name\_\_ == "\_\_main\_\_":  print("🚀 Main block chal gaya")  output  👀 Main file start  🚀 Main block chal gaya | main.py file:  import pdf\_reader  pdf\_reader.say\_hello()  👀 Main file start  Hello from PDF Reader! |

✅ Dekha? 🚀 Main block **nahi chala**, kyunki file import hui thi — **direct run nahi hui thi.**

|  |
| --- |
| agent = PDFReaderAgent("Class\_5.pdf") |

### 🔹 Line 2: agent = PDFReaderAgent("Class\_5.pdf")

📌 **Iska matlab kya hai?**  
Hamne PDFReaderAgent class ka object banaya jiska naam agent rakha.

📄 "Class\_5.pdf" is class ke constructor \_\_init\_\_() me ja raha hai aur PDF file ka naam pass ho raha hai.

🧠 Simple words me:

* agent ab wo helper AI hai jo PDF ko samajh chuka hai.
* Is object ke andar **self.pdf\_text** me poori PDF ka text store hai.

|  |
| --- |
| question = input("📄 PDF se kya poochna hai? Sawal likho: ") |

### 🔹 Line 3: input(...)

Yeh line user se input maang rahi hai. Jo bhi sawal tum type karogi, wo question variable me store ho jayega.  
Agar tum input do: What is Python?  
To question = "What is Python?" ban jayega.

|  |
| --- |
| answer = agent.ask(question) |

### 🔹 Line 4: answer = agent.ask(question)

Ham apne agent object ka ask() method call kar rahe hain, jisme tumhara question diya gaya.

🧠 Inside story:

* ask() method ek **prompt banata hai** jisme tumhara question aur PDF ka text combine hota hai.
* Phir wo Gemini model ko call karta hai: model.generate\_content(...).
* Gemini ka response return hota hai.

|  |
| --- |
| print("\n🤖 Agent ka jawab:")  print(answer) |

### 🔹 Line 5 & 6: print(...)

Bas Gemini ka response (jo answer variable me aya hai) usay print kar diya jata hai.

### ✅ Step-by-Step Summary of Code:

#### 🔹 **Step 1: Load API Key**

* load\_dotenv() → .env file se environment variable load karta hai.
* genai.configure(...) → Gemini ke API key ko configure karta hai.

#### 🔹 **Step 2: Load Gemini Model**

* genai.GenerativeModel(...) → Gemini model load karta hai. Yahan "gemini-2.0-flash" version use ho raha hai.
* model ek variable hai jo Gemini model ka object ban gaya.

#### 🔹 **Step 3: Define PDFReaderAgent Class**

* \_\_init\_\_ → class ka constructor, pdf\_path accept karta hai aur self.pdf\_text me text store karta hai.
* \_extract\_pdf\_text() → FitZ (PyMuPDF) se PDF ka sara text extract karta hai.
* ask() → Gemini model ko question + PDF ka content bhejta hai aur jawab return karta hai.

#### 🔹 **Step 4: Use the Agent**

* Agar \_\_name\_\_ == "\_\_main\_\_" ho, to user se question liya jata hai.
* Agent banaya jata hai (PDFReaderAgent("Class\_5.pdf")) aur ask() method se answer liya jata hai.

#### 🔹 **Step 5: Final Output**

* print() statement se Gemini ka jawab terminal me show hota hai.