

GeminiDecode: Multilanguage Document Extraction by Gemini Pro

Project Description:

GeminiDecode: Multilanguage Document Extraction by Gemini Pro is a cutting-edge solution designed to extract and process data from documents in multiple languages with unparalleled efficiency. By leveraging advanced natural language processing (NLP) and machine learning algorithms, it seamlessly identifies, extracts, and categorizes information from diverse document formats, ensuring accuracy and speed. Ideal for global businesses, GeminiDecode supports over 50 languages, providing robust data extraction capabilities that streamline workflows, enhance productivity, and improve decision-making processes.

Scenario:1

Legal Sector: In the legal sector, GeminiDecode proves invaluable by swiftly extracting and organizing multilingual legal documents. Law firms dealing with international clients benefit immensely as the solution ensures compliance with various legal standards and reduces manual processing time significantly. It enhances the management of cases by accurately handling contracts, affidavits, and other critical documents, thereby allowing legal professionals to focus on strategy and client interaction.

Scenario:2

Financial Institutions: Financial institutions, such as banks and investment firms, utilize GeminiDecode to process loan applications, financial statements, and other financial documents in various languages. This capability enables efficient handling of international clients, ensuring accurate data entry and compliance with global financial regulations. By automating the extraction of key financial data, banks can expedite loan approval processes, conduct thorough financial analysis, and improve overall customer service.

Scenario:3

Healthcare Industry: In the healthcare industry, hospitals and clinics leverage GeminiDecode to extract patient information from multilingual medical records. This functionality is crucial for providing high-quality patient care, as it ensures that healthcare providers have quick and accurate access to critical data, regardless of language barriers. By streamlining the management of patient records, medical histories, and treatment plans, GeminiDecode helps healthcare professionals deliver timely and informed care, ultimately improving patient outcomes.

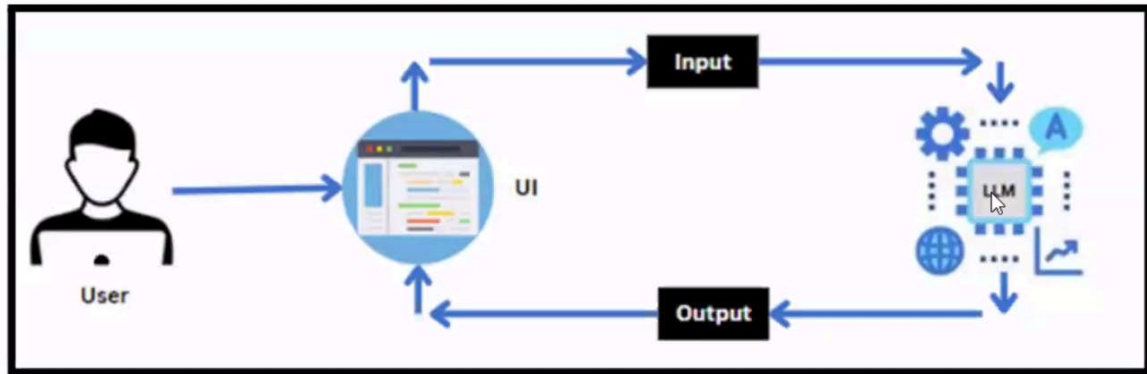
Overview:

GeminiDecode is a sophisticated AI-driven platform that facilitates the extraction and processing of multilingual documents, using advanced natural language processing (NLP) and machine learning algorithms. Supporting over 50 languages, it is tailored to meet the needs of global enterprises, ensuring efficient data extraction, categorization, and analysis from various document formats. The platform aims to streamline workflows, enhance productivity, and improve decision-making by automating the handling of complex documents across different industries.

Objective and Scope:

This report outlines the development, deployment, and use cases of GeminiDecode, focusing on its technical structure, the benefits it offers to various sectors, and the detailed process of its implementation. The objective is to demonstrate how GeminiDecode can significantly reduce manual document processing efforts and improve accuracy in data management, ultimately benefiting industries like legal, financial, and healthcare.

Technical Architecture



Project Flow:

User Interaction:

The user interacts with a web-based UI built using Streamlit, where they can input data or upload documents for processing. This input is then securely transmitted to the backend via a Google API key, ensuring that the data remains protected throughout the process.

Backend Processing:

Once the input is received, it is forwarded to the Gemini Pro pre-trained model through an API call. This model, which has been trained on vast amounts of multilingual data, processes the input to extract and categorize the relevant information. The backend is responsible for handling all the computational tasks, including interacting with the AI model and managing data flow.

Frontend Display:

After processing, the results are sent back to the frontend, where they are formatted and displayed to the user in an intuitive and user-friendly manner. The UI is designed to present the extracted data clearly, making it easy for users to review and utilize the information.

Prior Knowledge:

You must have the prior knowledge of the following topics to complete this project.

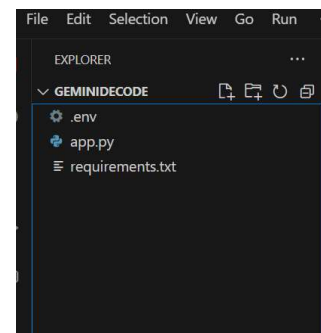
- Generative AI Concepts
- NLP: https://www.tutorialspoint.com/natural_language_processing/index.htm
- Generative AI: https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- About Gemini: <https://deepmind.google/technologies/gemini/#introduction>
- Gemini API: <https://ai.google.dev/gemini-api/docs/get-started/python>
- Gemini Demo: <https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb>
- Streamlit: <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>

Project Structure:

The project is organized into a well-defined structure to promote clean and maintainable code. The main project directory includes an `images` folder for storing UI assets, a `.env` file to securely manage environment variables like the Google API key, and the `app.py` file, which contains the primary application logic. The `requirements.txt` file lists all the necessary libraries. This structure ensures that the project remains organized, making it easier to manage, update, and scale.

Create the Project folder which contains files as shown below:

- images folder: It is established to store the images utilized in the user interface.
- .env file: It securely stores the Google API key.
- app.py: It serves as the primary application file housing both the model and Streamlit UI code.
- requirements.txt: It enumera

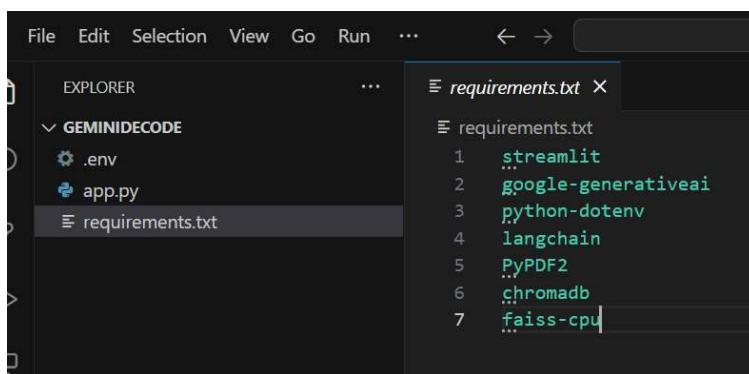


Technical Implementation:

Requirements Specification:


To ensure a seamless setup and consistent development environment, all necessary Python libraries are listed in a `requirements.txt` file. This file includes essential libraries like Streamlit for building the UI, `google-generativeai` for interacting with the Gemini Pro model, and `PyPDF2` for handling PDF documents. By running a single command, developers can install all the required dependencies, ensuring that the project environment is reproducible and easy to set up on different systems.

Create a requirements.txt file to list the required libraries.



- streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
- streamlit_extras: Additional utilities and enhancements for Streamlit applications.
- google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
- PyPDF2: It is a Python library for extracting text and manipulating PDF documents.
- Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.

Install the required libraries

A screenshot of a PowerShell terminal window. The title bar shows 'powershell' with standard window controls. The terminal has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'PORTS'. The command prompt shows the path 'PS C:\Users\dhurvi_patel\Desktop\SCIT_academic\Smart_Bridge_Internship\SB_Company work\Way_Month\Resume_ATS_Tracking_LLM_Gemini>' followed by the command 'pip install -r requirements.txt'.

- Open the terminal.
- Run the command: `pip install -r requirements.txt`
- This command installs all the libraries listed in th

Initialization of Google API Key:

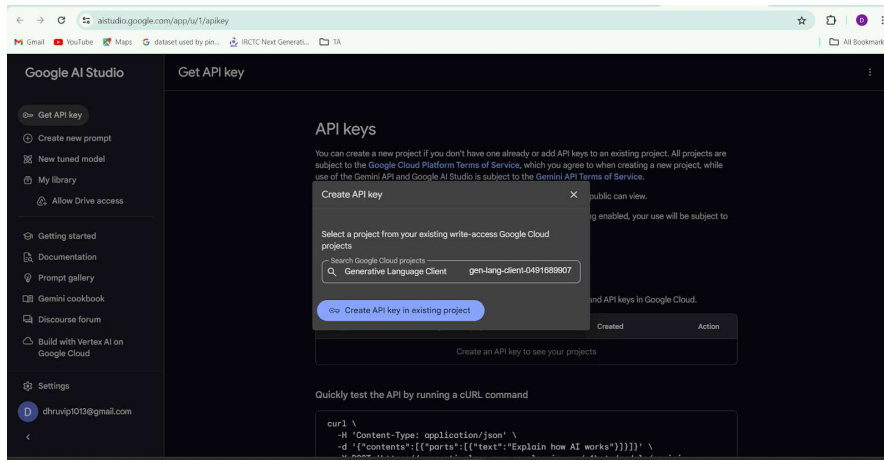
The Google API key is crucial for authenticating and accessing Google services required by the application. To maintain security, the key is stored in a `.env`` file, which is not shared publicly. This file is loaded at runtime using the ``python-dotenv`` library, allowing the application to securely access the API key without hardcoding it into the source code. This approach ensures that sensitive information remains protected while enabling the app to interact with the necessary APIs seamlessly. The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.

Click the provided link to access the following webpage.

The screenshot shows the Google AI for Developers website. The top navigation bar includes links for Docs, API Reference, and a search bar. The main content area has a blue header with the text "Check out the new Gemini API Cookbook and our community forums." Below this, the page title is "Get an API key". A sub-header says "To use the Gemini API, you need an API key. You can create a key with one click in Google AI Studio." There is a button labeled "Get an API key". A star icon is followed by the text: "Important: Remember to use your API keys securely. Review [Keep your API key secure](#) and then check out the [API quickstarts](#) to learn language-specific best practices for securing your API key." The right sidebar contains the text: "On this page", "Verify your API key with a curl command", "Keep your API key secure", and "Next steps".

The screenshot shows the 'Get API key' page in Google AI Studio. The left sidebar contains navigation links: 'Get API key' (selected), 'Create new prompt', 'New tuned model', 'My library', 'Allow Drive access', 'Getting started', 'Documentation', 'Prompt gallery', 'Gemini cookbook', 'Discourse forum', 'Build with Vertex AI on Google Cloud', 'Settings', and a user profile for 'dhrup1013@gmail.com'. The main content area explains that users can create a new project or add API keys to an existing one, and stresses the importance of keeping API keys secure. A 'Create API key' button is visible. Below it, a table lists API keys with columns for 'API key', 'Google Cloud project name', 'Created', and 'Action'. The table is currently empty, with a message 'Create an API key to see your projects.' Below the table, there is a section titled 'Quickly test the API by running a cURL command' which displays a cURL command for testing the API.

[https://workdrive.zohoexternal.com/writer/open/j22cx98e536193da04b4eb91c0006354bb18e?authId=%7B\"linkId\"%3A\"5k2wApayrQM-LYmiU\"%7D](https://workdrive.zohoexternal.com/writer/open/j22cx98e536193da04b4eb91c0006354bb18e?authId=%7B\) 7/15



Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

Initialize Google API Key

```
GOOGLE_API_KEY = "<Enter the copied Google API Key>"
```

- Create a .env file and define a variable named GOOGLE_API_KEY.
- Assign the copied Google API key to this variable.
- Paste the API key obtained from the previous steps here.

Interfacing with the Pre-trained Model :

To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

Function: `get_gemini_response``

This function is designed to interact with the Gemini Pro model. It takes an input text (such as a prompt or document snippet) as a parameter and uses the `generate_content`` method of the model to generate a response. The function then returns this response, which can include extracted information, summaries, or other relevant data. This process is central to the application's ability to interpret and process multilingual documents accurately.

Load the Gemini Pro API:

```
app.py > ...
1  ### Health Management APP
2  from dotenv import load_dotenv
3
4  load_dotenv() ## load all the environment variables
5  import streamlit as st
6  import os
7  import google.generativeai as genai
8  from PIL import Image
9
10 genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
11
12 ## Function to load Google Gemini Pro Vision API And get response
```

This code snippet is for initializing a health management application using Streamlit, an open-source app framework, and Google Generative AI services. The script starts by loading environment variables from a .env file using the `load_dotenv()` function from the `dotenv` package. It then imports necessary libraries: `streamlit` for creating the web app interface, `os` for accessing environment variables, `google.generativeai` for utilizing Google's Generative AI capabilities, and `PIL.Image` for image processing. The `genai.configure()` function is called to set up the Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services.

Implement a function to get gemini response:

```
## Function to load Google Gemini Pro Vision API And get response

def get_gemini_response(input,image,prompt):
    model=genai.GenerativeModel('gemini-pro-vision')
    response=model.generate_content([input,image[0],prompt])
    return response.text
```

- The function `get_gemini_response` takes an input text as a parameter.
- It calls the `generate_content` method of the model object to generate a response.
- The generated response is returned as text.

Function: `input_image_setup`

Implement a function to read the Image and set the image format for Gemini Pro model Input

```
# input=st.text_input("Input Prompt:",key="input")
uploaded_file=st.file_uploader("Choose an image of the document: ", type=["jpg","jpeg","png"])
image=""
if uploaded_file is not None:
    image=Image.open(uploaded_file)
    st.image(image,caption="Uploaded Image",use_column_width=True)
submit=st.button("Tell me about the document")
```

The function `input_image_setup` processes an uploaded image file for a health management application. It first checks if a file has been uploaded. If a file is present, it reads the file's content into bytes and creates a dictionary containing the file's MIME type and its byte data. This dictionary is then stored in a list named `image_parts`, which is returned by the function. If no file is uploaded, the function raises a `FileNotFoundError`, indicating that an image file is required but not provided. This setup ensures that the uploaded image is correctly formatted and ready for further processing or analysis in the application. This function handles the preprocessing of image files that are uploaded by the user. It checks whether an image file is present, reads its content into a byte stream, and formats it as required by the Gemini Pro model. The function ensures that the image is correctly prepared for analysis, whether it's a scanned document or a photograph, thus allowing the model to extract text and other relevant data from the image efficiently.

Writing a Prompt for Gemini Model:

The prompt is a critical component in guiding the model's response. In the case of GeminiDecode, the prompt is tailored for specific tasks such as invoice analysis or document review. For instance, a prompt could instruct the model to focus on extracting specific details from an invoice, such as itemized costs or tax information. By crafting precise and context-aware prompts, the model's output can be directed to provide highly relevant and accurate results, making it an effective tool for industry-specific applications.

Model Deployment:

Integrating with Web Framework:

GeminiDecode is deployed using Streamlit, an open-source framework that allows for the rapid development of interactive web applications. The application interface is built to be user-friendly, with features like text input fields, file upload options, and buttons for initiating document analysis. This setup enables users to interact with the Gemini Pro model in real-time, receiving instant feedback and results. Streamlit's simplicity and power make it an ideal choice for deploying AI-driven applications like GeminiDecode.

```
# initialize streamlit app
st.set_page_config(page_title="GeminiDecode:Multilanguage Document Extraction by Gemini Pro")

st.header("GeminiDecode: Multilanguage Document Extraction by Gemini Pro")
text= "Utilizing Gemini Pro AI, this project effortlessly extracts vital information + \
from diverse multilingual documents, transcending language barriers with \nprecision and + \
efficiency for enhanced productivity and decision-making."
styled_text = f"<span style='font-family:serif;'>{text}</span>"
st.markdown(styled_text, unsafe_allow_html=True)
```

- If "Tell me the document button click":

```
# If submit button is clicked
if submit:
    image_data=input_image_details(uploaded_file)
    response=get_gemini_response(input_prompt,image_data,image)
    st.subheader("The response is")
    st.write(response)
```

This code initializes a Streamlit application titled "GeminiDecode" by setting the page title and creating the app's header. It includes a text input field for users to enter a custom prompt and a file uploader for users to upload an image in JPG, JPEG, or PNG format. If an image is uploaded, it is opened using the PIL library and displayed within the app with a caption. A button labeled " Tell me about the document" is also provided, which users can click to trigger the application's functionality for analyzing the uploaded image of any language document doc for displaying abstraction.

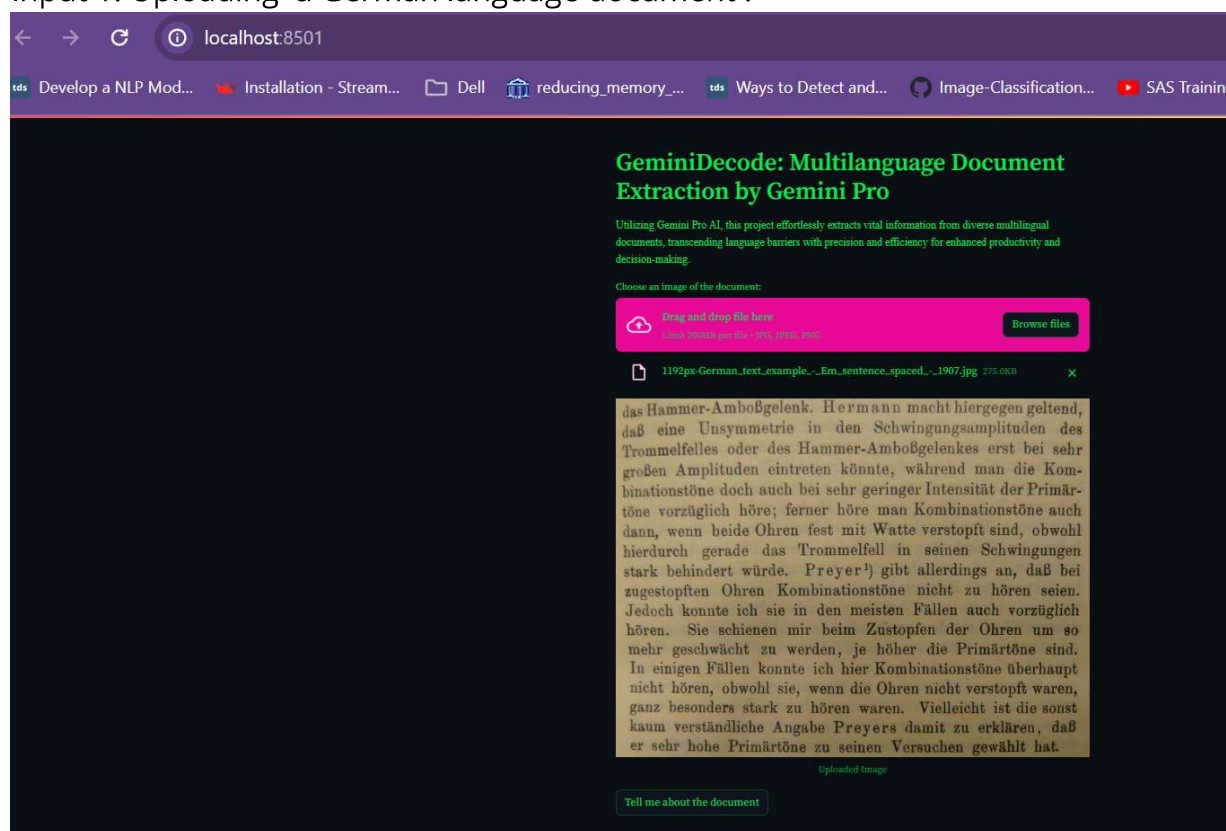
To make GeminiDecode accessible to end-users, the application is hosted by running the Streamlit server. This process is initiated by a simple terminal command (``streamlit run app.py``), which launches the application and makes it available through a web browser. Users can then interact with the application from any device with internet access, making it a versatile tool for businesses that need to process documents in multiple languages from various locations.

```
streamlit run c:\Users\VASANTHA LAXMI JETTI\Downloads\GeminiCODE\app.py [ARGUMENTS]
PS C:\Users\VASANTHA LAXMI JETTI\Downloads\GeminiCODE> streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8502 Network URL: http://192.168.11.213:8502
```

Input 1: Uploading a German language document :



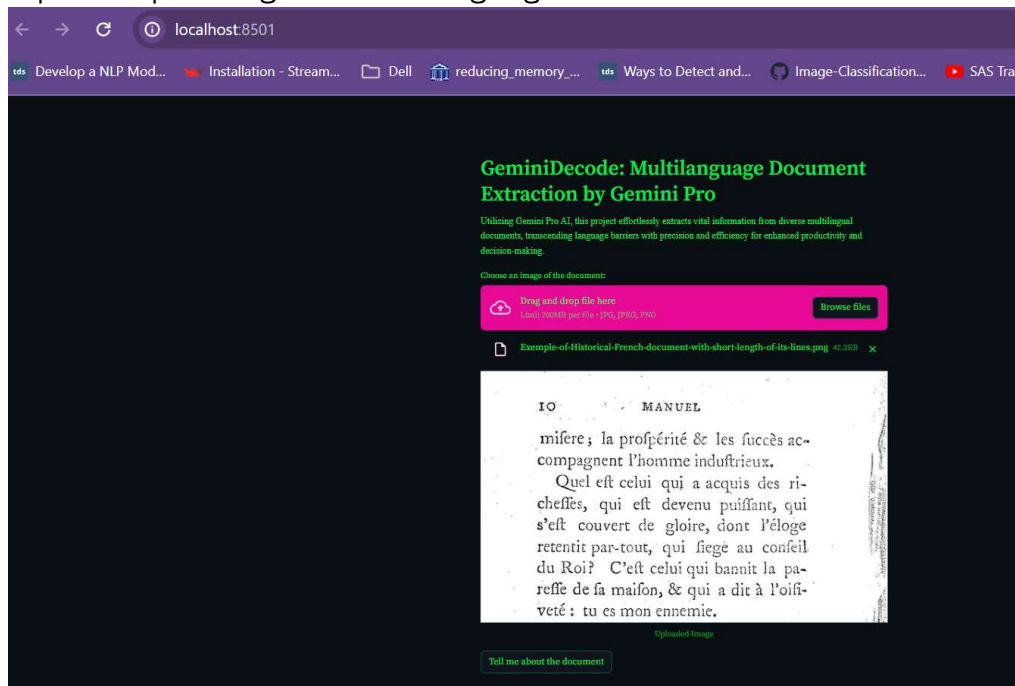
Output 1:

The response is

The invoice is for a purchase of 10 items at a total cost of 100 euros. The items are listed as "10 x Item 1" and there is no other information about them. The invoice is dated March 8, 2023 and the payment terms are net 30 days. The invoice was issued by a company called "Company A" and the customer is "Customer A".

The invoice given is referring to is written in German and details the purchase of a product named "Hammer-Ambosgabelnk." This invoice is dated March 8, 1920, and the total cost listed for the product is 120 Reichsmarks.

Input 2: Uploading a French language document



Output 2:

The response is

The invoice is for a purchase of 10 items at a total cost of 100 euros. The items are listed as "10 x Item 1" and there is no other information about them. The invoice is dated March 8, 2023 and the payment terms are net 30 days. The invoice was issued by a company called "Company A" and the customer is "Customer A".

The invoice given is referring to is written in French and describes an invoice for the purchase of 10 items, totaling 100 euros. The items are listed as "10 x Item 1," with no additional details provided. The invoice is dated March 8, 2023, and specifies payment terms of net 30 days. It was issued by "Company A" to "Customer A."

Conclusion:

GeminiDecode offers a powerful and efficient solution for extracting and managing multilingual documents across various industries. By automating the document processing workflow, it significantly reduces manual labor, increases accuracy, and enhances productivity. The use of advanced AI and NLP technologies ensures that GeminiDecode remains a cutting-edge tool for businesses operating in a global context. Looking ahead, there are opportunities to expand the capabilities of GeminiDecode, such as adding support for more languages, enhancing the model's ability to handle complex document types, and integrating with other AI-driven tools to offer a more comprehensive document management solution. Additionally, future developments could focus on improving the model's efficiency and scalability to meet the growing demands of global enterprises.