**How to Install Python**

1. Visit the official Python website: <https://www.python.org/downloads>.
2. Download and install the latest version of Python compatible with your operating system.
3. During installation, ensure you check the option to "Add Python to PATH" for easy command-line usage.

**Install Dependencies for Article Generator in a Virtual Environment**

1. Open your terminal or command prompt and navigate to your project directory.
2. Run the following command to create a virtual environment:

python -m venv venv

1. Activate the virtual environment every time you run the project:
   * On Windows:

venv\Scripts\activate

* + On macOS/Linux:

source venv/bin/activate

**Install Required Packages**

With the virtual environment activated, run the following commands to install the necessary dependencies:

1. **Install Streamlit**

pip install streamlit

1. **Install Python Dotenv**

pip install python-dotenv

1. **Install Google Generative AI**

pip install google-generativeai

**Run the Article Generator Project**

Run the following command to start the project:

streamlit run article.py

**Packages Installed with Streamlit**

The following packages are installed as dependencies when you install Streamlit:

1. **pytz**
2. **watchdog**
3. **urllib3**
4. **tzdata**
5. **typing-extensions**
6. **tornado**
7. **toml**
8. **tenacity**
9. **smmmap**
10. **six**
11. **rpds-py**
12. **pygments**
13. **pyarrow**
14. **protobuf**
15. **pillow**
16. **packaging**
17. **numpy**
18. **narwhals**
19. **mdurl**
20. **MarkupSafe**
21. **idna**
22. **colorama**
23. **charset-normalizer**
24. **certifi**
25. **cachetools**
26. **blinker**
27. **attrs**
28. **requests**
29. **referencing**
30. **python-dateutil**
31. **markdown-it-py**
32. **jinja2**
33. **gitdb**
34. **click**
35. **rich**
36. **pydeck**
37. **pandas**
38. **jsonschema-specifications**
39. **gitpython**
40. **jsonschema**
41. **altair**
42. **streamlit**

Explanation of the code

|  |
| --- |
| import streamlit as st  import google.generativeai as genai  import os  from dotenv import load\_dotenv  # Load environment variables from the .env file  load\_dotenv()  # Access the API key from the environment  API\_KEY = os.getenv("GEMINI\_API\_KEY")  print(f"Loaded API Key: {API\_KEY}")  if API\_KEY:  # Configure the Gemini API  genai.configure(api\_key=API\_KEY)  else:  st.error("API key is not set. Please configure the GEMINI\_API\_KEY environment variable.")  def generate\_article(keywords, writing\_style, word\_count):  """  This function uses the Gemini API to generate an article.  Args:  keywords (str): Keywords for the article.  writing\_style (str): Desired writing style (e.g., Formal, Informal).  word\_count (int): Target word count for the article.  Returns:  str: The generated article text.  """  try:  # Create the prompt for text generation  prompt = f"Write a {writing\_style} article of approximately {word\_count} words about {keywords}."  model = genai.GenerativeModel('gemini-pro')  # Call the text generation API  response = model.generate\_content(prompt)    # Extract the generated text from the response    return response.text  except Exception as e:  st.error(f"An error occurred: {e}")  return ""  # Streamlit interface  st.title("Article Generator")  keywords = st.text\_input("Enter keywords:")  writing\_style = st.selectbox("Choose writing style:", ["Formal", "Informal", "Technical", "Narrative"])  word\_count = st.slider("Select word count:", 100, 2000)  if st.button("Generate"):  if not API\_KEY:  st.error("API key is not set. Please configure the GEMINI\_API\_KEY environment variable.")  else:  article = generate\_article(keywords, writing\_style, word\_count)  if article:  st.write(article)  st.download\_button(label="Download Article", data=article,file\_name='article.txt', mime="text/txt") |