



AI Email Generator



Table of Contents

- Introduction
- Design
- Implementation
- Test
- Improvement Suggestions
- Conclusion

Introduction

- Generating a Customer Email Project
- Design and implement a system leveraging OpenAI's capabilities to generate content based on user input.
- Frontend: Next.js
- Backend: Python scripts and OpenAI API.

```
▼ nodeproj
  > __pycache__
  > .next
  > node_modules
  ▼ pages
    > api
    JS index.js
    # index.module.css
  > public
  ⚙ .env
  💎 .gitignore
  🌀 answer.py
  {} package-lock.json
  {} package.json
  🌀 products.py
  ⓘ readme.md
  ≡ requirements.txt
```

Language English ▾

The TechPro Ultrabook is a sleek and lightweight option for everyday use, with its 13.3-inch display, 8GB RAM, 256GB SSD, and Intel Core i3 processor.



Design

- **Frontend:** Built using the Next.js framework and React library. Provides the user interface for input and displaying results.
- **Backend:** Python scripts that interface with the OpenAI API. The main logic for generating content.
- **Data Flow:**
 - User inputs question and selects language on the frontend.
 - Backend receives input, processes it, and interacts with OpenAI.
 - Resulting content is sent back to the frontend for display.

Implementation (Frontend)

- Framework: Next.js – a React framework.
- Main Components:
 - Language Selector: Dropdown menu that lets users choose between English, Chinese, or Spanish.
 - Question Box: Textarea for inputting or viewing the default customer comment.
 - Answer Box: Area where the OpenAI-generated content is displayed.
 - Submit Button: Initiates the content generation process.
 - Loading Spinner: Indicates data fetching in progress.
- Styling: Custom CSS styles from index.module.css for layout and aesthetics.

```
export const [ 'getStaticProps', 'getStaticPaths' ] = [ () => {
  const locales = ['en', 'zh', 'es'];
  const defaultLocale = 'en';
  return {
    props: {
      locales,
      defaultLocale,
    },
  };
}, () => {
  return {
    paths: locales.map((locale) => ({ locale })),
    fallback: false,
  };
} ];
```

Language

Generating a Question...

✓ English
Chinese
Spanish

```
export const handleSubmit = async (e) => {
  e.preventDefault();
  const { question, language } = e.target.elements;
  const data = await fetch('/api/generate-question', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ question, language }),
  });
  const json = await data.json();
  if (json.status === 'error') {
    throw new Error(json.message);
  }
  const { answer } = json;
  setAnswer(answer);
  setQuestion('');
  setLanguage(language);
};
```

Implementation (Backend)

- OpenAI Integration: The system uses the official OpenAI library for Python. API calls are made to generate content based on prompts.
- Main Logic in answer.py:
 - generate_customer_comment(): Creates a product comment using OpenAI.
 - generate_email_subject(): Derives an email subject from a customer comment.
 - generate_summary_of_comment(): Summarizes a comment.
 - sentiment_analysis(): Assesses sentiment of a comment.
 - generate_email(): Constructs a full email from a comment, its sentiment, summary, and a subject.
- Product Data in products.py: Contains a dictionary of product details used as a foundation for generating comments.

```
import os
import openai
from dotenv import load_dotenv
import sys

from products import products
import time

load_dotenv()
openai.api_key = os.getenv("OPENAI_API_KEY")

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        # stop is the sequence of characters the
        # model should stop generating
        temperature=0.7,
        max_tokens=300
    )
    return response.choices[0].message["content"]

def generate_customer_comment():
    prompt = """
    The following text is the products' descriptions. Please generate a 300 words maximum comment about the products.

    """
    products_descriptions = (products)

    PS: The comment should have a maximum of 300 words.
    """
    return get_completion(prompt)

def generate_email_subject(comment, language):
    prompt = """
    Assuming that you provide customer support for an electronic product company.
    The following text is the customer's comment about the products. Please generate an email subject in {language} for the following comment. The subject will be used as the subject of the email to be sent to the customer.

    Customer's Comment: {comment}
    """
    return get_completion(prompt)

def generate_summary_of_comment(language, comment):
    prompt = """
    Assuming that you provide customer support for an electronic product company.
    The following text is the comment of products. Please generate a summary of the comment in {language}.

    """
    (comment)
    """
    return get_completion(prompt)

def sentiment_analysis(comment):
    prompt = """
    Assuming that you provide customer support for an electronic product company.
    Please do sentiment analysis based on the following comment.
    The result of the sentiment analysis shows whether the customer's comment is Positive or Negative.

    """
    (comment)
    """
    return get_completion(prompt)

def generate_email(language, comment):
    subject = generate_email_subject(comment, language)
    summary = generate_summary_of_comment(language, comment)
    sentiment = sentiment_analysis(comment)

    prompt = """
    Assuming that you provide customer support for an electronic product company.
    Please create an email in {language} to be sent to the customer based on:
    1. The customer's comment: {comment}
    2. The summary of the customer's comment: {summary}
    3. The result of the sentiment analysis of the customer's comment: {sentiment}
    4. The Subject of the email: {subject}

    PS: The email should have 200 words maximum.
    """
    return get_completion(prompt)

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Error: No action provided")
        sys.exit()

    action = sys.argv[1]

    # Debugging statement to print the received action

    if action == "generate_customer_comment":
        print(generate_customer_comment())
    elif len(sys.argv) == 3:
        # Ensure there are enough arguments for other actions
        comment = sys.argv[2]
        language = sys.argv[1]
        print(generate_email(language, comment))
    else:
        print("Error: Not enough arguments provided")
```

Test

- Initial Load: On the first load, the system fetches a default comment to display.
- User Interaction: On submission, the frontend communicates with the `/api/generateAnswer` endpoint, passing the user's question and selected language.
- Backend Processing: Python script (`answer.py`) is executed, and the OpenAI API is queried.
- Result Display: The frontend then displays the generated content in the answer box.

```
abderrahmane@Abderrahmanes-MacBook-Pro nodeproj % npm run dev

> openai-qa@0.1.0 dev
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
info - Loaded env from /Users/abderrahmane/Desktop/ai-projects/nodeproj/.env
event - compiled client and server successfully in 192 ms (150 modules)
wait - compiling / (client and server)...
event - compiled client and server successfully in 49 ms (170 modules)
wait - compiling /api/generateAnswer (client and server)...
event - compiled successfully in 15 ms (46 modules)
Executing command: python3 answer.py generate_comment
API resolved without sending a response for /api/generateAnswer, this may result in stalled requests.
wait - compiling /_error (client and server)...
event - compiled client and server successfully in 52 ms (182 modules)
warn - Fast Refresh had to perform a full reload. Read more: https://nextjs.org/docs/messages/fast-refresh-reload
```

Language English ▾

Generating a Question...

Subject: Discover the Perfect Laptop for Your Needs and Budget Dear [Customer's Name], Thank you for your positive feedback on our range of laptops! We appreciate your interest in our TechPro Ultrabook, BlueWave Gaming Laptop, and PowerLite Convertible. We are thrilled to offer a variety of options that cater to different needs and budgets in the world of computers and laptops. The TechPro Ultrabook, as you mentioned, is a sleek and lightweight option ideal for everyday use. With its 13.3-inch display, 8GB RAM, 256GB SSD, and Intel Core i5 processor, it strikes the perfect balance between performance and portability. Whether you're working on-the-go or enjoying multimedia content, this Ultrabook is designed to meet your needs. For gaming enthusiasts, our BlueWave Gaming Laptop is the ultimate choice. With its 15.6-inch display, 16GB RAM, 512GB SSD, and NVIDIA GeForce RTX 3060, it delivers a truly immersive gaming experience. From high-performance graphics to seamless gameplay, this



Improvement Suggestions

- Expand Language Options: Incorporate more languages for broader user reach.
- Advanced Error Handling: Implement better error handling for edge cases or unexpected OpenAI API behaviors.
- User Customizations: Allow users to specify the type and style of content they want.
- Optimize for Mobile: Ensure the platform is fully responsive for mobile users.



Conclusion

- Achievements: Successfully built a system that leverages OpenAI to generate email content based on user input and product details.
- Applications: Can be extended to other domains such as customer support, content generation, marketing, and more.



Bibliography & Github Link

- https://github.com/Alami64/cs589_week3_hw2_customer_email
- https://hc.labnet.sfbu.edu/~henry/sfbu/course/deeplearning_ai/chatgpt_prompt_eng_for_developer/slide/exercise_chatgpt_prompt_eng_for_developer.html