10월

시도1

```
useEffect(() => {
  const fetchData = async () => {
    try {
      const result = await axios.post(
        'https://api.openai.com/v1/chat/completions',
        {
          model: 'gpt-4o', // 사용하려는 모델 (GPT-4o 기준)
          messages: [{role: 'user', content: prompt}],
          max_tokens: 200, // 생성할 텍스트의 길이
        },
        {
          headers: {
            'Content-Type': 'application/json',
            Authorization: Bearer ${apiKey}, // API 키를 헤더
          },
        }
      );


      setResponse(result.data.choices[0].message.content.tr
    } catch (error) {
      console.error('Error calling OpenAI API:', error);
      setResponse('Error occurred while generating text.');
    } finally {
      setLoading(false);
    }
  };
```

```
    fetchData();
  }, [prompt]);
```

결과

404 에러 뜸 (위치나 설정 에러인 것 같음)

시도2

```
  useEffect(() => {
    const fetchData = async () => {
      try {
        const result = await axios.post(
          'https://api.openai.com/v1/chat/completions',
          {
            model: 'gpt-4o-mini', // 사용하려는 모델 (GPT-4o 기준
            messages: [{role: 'user', content: prompt}],
            max_tokens: 200, // 생성할 텍스트의 길이
          },
          {
            headers: {
              'Content-Type': 'application/json',
              Authorization: Bearer ${apiKey}, // API 키를 헤더
            },
          }
        );


        setResponse(result.data.choices[0].message.content.tr
      } catch (error) {
        console.error('Error calling OpenAI API:', error);
        setResponse('Error occurred while generating text.');
      } finally {
        setLoading(false);
      }
    };

    fetchData();
```

결과

429 에러 → 너무 많은 요청

시도3

```
useEffect(() => {
  const fetchData = async () => {
      let retryAttempts = 3; // 최대 재시도 횟수
      let retryDelay = 1000; // 초기 대기 시간 (1초)

      while (retryAttempts > 0) {
          try {
              const result = await axios.post(
                  'https://api.openai.com/v1/chat/completio
                  {
                      model: 'gpt-4o-mini', // 사용하려는 모델
                      messages: [{ role: 'user', content: p
                      max_tokens: 200, // 생성할 텍스트의 길이
                  },
                  {
                      headers: {
                          'Content-Type': 'application/json
                          Authorization: `Bearer ${apiKey}`
                      },
                  }
              );

              setResponse(result.data.choices[0].message.co
              break; // 요청이 성공하면 반복 종료
          } catch (error) {
              if (error.response && error.response.status =
                  console.error('Too many requests. Retryin
                  retryAttempts--;
                  await new Promise(resolve => setTimeout(r
                  retryDelay *= 2; // 대기 시간 증가
              } else {
                  console.error('Error calling OpenAI API:'
                  setResponse('Error occurred while generat
```

```
                    break; // 다른 오류가 발생하면 반복 종료
                }
            } finally {
                setLoading(false);
            }
        }
    };

    fetchData();
}, [prompt]);
```

결과

3번 시도하긴 하는데 한 번 시도마다 429에러 뜸

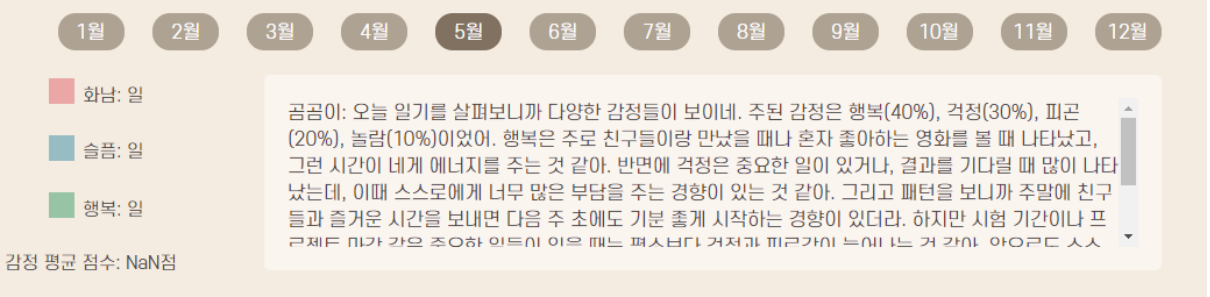
프롬프트가 너무 긴가..? (아니었음)

딜레이 주기 → 실패

해결!

해결 방법

1. GPT를 사용해 해결되지 않았던 걸 계속 검색해봤는데 해결하지 못함
2. 구글에 Too many Qequest를 검색해서 같은 사례를 검색
3. 카드 등록하고 key를 새로 등록받으면 된다는 사례 발견
4. 카드 등록으로 5.5달러 결제(ㅠ) 후 key 재발급
5. 성공!



문제(사소한) 및 느낀점

1. 새로고침할 때마다 요청(돈이 계속 든다)
2. 하지만 비용이 사소함 (한 번 요청 당 10원)
3. GPT-4o의 성능이 확실히 더 좋음 (mini 보다)
4. 이젠 여러 대화 또는 한 일기를 받아서 GPT한테 보내주는 걸 해야할 것 같음(서버가 완료되면)
5. 이제 모달을 제작해야겠다
6. 프롬프트에서 분석을 비율(%)로 해달라고 요청했는데 더 좋은 것 같음

최종 코드

```
  useEffect(() => {
    const fetchData = async () => {
      if(ex_diary=='') {
        setResponse('오늘의 이야기를 말해주면 분석해줄 수 있어! 언제든
        return;
      }

      const prompt = `일기 내용: ${ex_diary}\n너는 일기를 녹음하는

      setLoading(true);

      try {
          const result = await axios.post(
              'https://api.openai.com/v1/chat/completions',
              {
                  model: 'gpt-4o', // 사용하려는 모델
                  messages: [{ role: 'user', content: prompt
                  max_tokens: null, // 생성할 텍스트의 길이
              },
              {
                  headers: {
                      'Content-Type': 'application/json',
                      Authorization: `Bearer ${apiKey}`, // A
                  },
              }
```

```
            );

            setResponse(result.data.choices[0].message.content.
        } catch (error) {
                console.error('Error calling OpenAI API:', erro
                setResponse('Error occurred while generating te
        } finally {
            setLoading(false);
        }
    };


    fetchData();

}, [ex_diary]);
```

할 일: 모달 추가 + 모달에 GPT 불러오기 → 모달 완성

발생한 문제 + 변경 사항

1. GPT가 프롬프트 말을 잘 안 듣고 줄바꿈을 제대로 안 해줌
2. 글씨체 변경(전에 사용했던 글씨체가 가독성이 좋지 않아 귀여운 폰트로 변경(네이버 중학생 폰트)
3. GPT의 응답을 받아서 후처리하는 것으로 변경했는데 응답을 받아오기 전에 후처리가 실행돼서 오류 발생(해결해야 함)

진행 상황

디자인 완료

```
@font-face {
  font-family: "MS";
  src: url("/MS.ttf") format("truetype");
  font-weight: normal;
}
pre {
  color: #292622;
  font-family: 'MS', serif;
  white-space: pre-wrap; /* 줄바꿈 유지 */
  max-width: 100%; /* 최대 너비 설정 */
}


.title1 {
  position: absolute;
  top: 23px;
  left: 233px;
  white-space: pre-wrap;
  text-align: center;
  color: #574D42;
}
.title2 {
  position: absolute;
  top: 23px;
```

```
  left: 794px;
  white-space: pre-wrap;
  text-align: center;
  color: #574D42;
}
.title3 {
  position: absolute;
  top: 23px;
  left: 1349px;
  white-space: pre-wrap;
  text-align: center;
  color: #574D42;
}

.diaryCheckPopupChild1
{
  position: absolute;
  top: 64px;
  left: 61px;
  border-radius: var(--br-3xs) var(--br-3xs) var(--br-3xs) va
  background-color: #FFF9F1;
  width: 495px;
  height: 666px;
  z-index: 0;
}
.diaryCheckPopupChild1_content{
  position: absolute;
  top: 107px;
  left: 108px;
  width: 387px;
  height: 486px;
  z-index: 0;
  color: #292622;
  object-fit: cover;
  font-size: 25px;
  font-family: 'MS', serif;}
.diaryCheckPopupChild2
{
```

```
  position: absolute;
  top: 64px;
  left: 619px;
  border-radius: var(--br-3xs) var(--br-3xs) var(--br-3xs) va
  background-color: #E0D2C0;
  width: 495px;
  height: 666px;
  z-index: 0;

}
.diaryCheckPopupChild2_content{
  position: absolute;
  top: 82px;
  left: 673px;
  width: 387px;
  height: 486px;
  z-index: 1;
  color: #292622;
  object-fit: cover;
  font-size: 25px;
  font-family: 'MS', serif;
}
.diaryCheckPopupChild3
{
  position: absolute;
  top: 64px;
  left: 1178px;
  border-radius: var(--br-3xs) var(--br-3xs) var(--br-3xs) va
  background-color: #FFF9F1;
  width: 495px;
  height: 666px;
  z-index: 0;
}
.diaryCheckPopupChild3_content{
  position: absolute;
  top: 90px;
  left: 1348px;
  width: 320px;
```

```
  height: 575px;
  z-index: 0;
  color: #292622;
  object-fit: cover;
  font-size: 22px;
  font-family: 'MS', serif;
}

.gomgom {
  position: absolute;
  top: 21px;
  left: 651px;
  width: 431px;
  height: 709px;
  z-index: 0;
  object-fit: cover;
}
.gomgom_book {
  position: absolute;
  object-fit: cover;
  top: 108px;
  left: 1215px;
  width: 100px;
  height: 91.18px;
}
.gomgom_music {
  position: absolute;
  object-fit: cover;
  top: 397px;
  left: 1199px;
  width: 130px;
  height: 130px;
}
.gomgom_movie {
  position: absolute;
  object-fit: cover;
  top: 242px;
  left: 1206px;
```

```
  width: 120px;
  height: 120px;
}
.gomgom_food {
  position: absolute;
  object-fit: cover;
  top: 551px;
  left: 1196px;
  width: 130px;
  height: 130px;
}
.book {
  position: absolute;
  object-fit: cover;
  top: 204px;
  left: 1258px;
  width: 15px;
  height: 18px;
  font-size: 15px;
  color: #4B443B;
}
.movie {
  position: absolute;
  object-fit: cover;
  top: 353px;
  left: 1251px;
  width: 29px;
  height: 18px;
  font-size: 15px;
  color: #4B443B;
}
.music {
  position: absolute;
  object-fit: cover;
  top: 507px;
  left: 1251px;
  width: 29px;
  height: 18px;
```

```
  font-size: 15px;
  color: #4B443B;
}
.food {
  position: absolute;
  object-fit: cover;
  top: 661px;
  left: 1251px;
  width: 29px;
  height: 19px;
  font-size: 15px;
  color: #4B443B;
}
.book_title,
.book_content,
.movie_title,
.movie_content,
.music_title,
.music_content,
.food_title,
.food_content{
  position: absolute;
  object-fit: cover;
  font-size: 16px;
  left: 1347px;
  width: 300px;
  color: #292622;
}

.book_content,
.movie_content,
.music_content,
.food_content{
  font-family: 'MS', serif;
  font-size: 22px;
}

.book_title{
```

```
  top: 120px;
}
.book_content{
  top: 140px;
}
.movie_title{
  top: 264px;
}
.movie_content{
  top: 284px;
}
.music_title{
  top: 430px;
}
.music_content{
  top: 450px;
}
.food_title{
  top: 585px;
}
.food_content{
  top: 605px;
}

.diaryCheckPopup {
  position: relative;
  width: 1734px;
  background-color: var(--color-linen-200);
  height: 765px;
  overflow: hidden;
  max-width: 100%;
  max-height: 100%;
  text-align: left;
  font-size: var(--font-size-5xl);
  color: #EEE8DB;
  font-family: var(--font-s-core-dream);
  border-radius: var(--br-3xs) var(--br-3xs) var(--br-3xs) va
```

```
}
```

곰곰이의 일기 불러오기 완료(GPT)

```
  useEffect(() => {
    const fetchData1 = async () => {
      if(ex_diary=='') {
        setResponse1('너의 하루를 들으면 일기를 전해주고 싶어져 :▷');
        return;
      }

      const prompt1 = `일기 내용: ${ex_diary}\n너는 일기를 녹음하는

      setLoading1(true);

      try {
          const result1 = await axios.post(
              'https://api.openai.com/v1/chat/completions',
              {
                  model: 'gpt-4o', // 사용하려는 모델
                  messages: [{ role: 'user', content: prompt1
                  max_tokens: null, // 생성할 텍스트의 길이
              },
              {
                  headers: {
                      'Content-Type': 'application/json',
                      Authorization: `Bearer ${apiKey}`, // A
                  },
              }
          );

          setResponse1(result1.data.choices[0].message.conten
      } catch (error) {
              console.error('Error calling OpenAI API:', erro
              setResponse1('Error occurred while generating t
      } finally {
          setLoading1(false);
```

```
      }
    };


    fetchData1();

}, []);
```

곰곰이의 추천 미완

```
useEffect(() => {
  const fetchData2 = async () => {
    if(ex_diary=='') {
      setResponse2('오늘 하루에 도움될만한 걸 추천해줄게 :)');
      return;
    }

    const prompt2 = `일기 내용: ${ex_diary}\n너는 일기를 녹음하는 일

    setLoading2(true);

    try {
        const result2 = await axios.post(
            'https://api.openai.com/v1/chat/completions',
            {
                model: 'gpt-4o', // 사용하려는 모델
                messages: [{ role: 'user', content: prompt2 }
                max_tokens: null, // 생성할 텍스트의 길이
            },
            {
                headers: {
                    'Content-Type': 'application/json',
                    Authorization: `Bearer ${apiKey}`, // API
                },
            }
        );

        setResponse2(result2.data.choices[0].message.content.
```

```
    } catch (error) {
          console.error('Error calling OpenAI API:', error.
          setResponse2('Error occurred while generating tex
    } finally {
        setLoading2(false);
    }
  };


  fetchData2();

}, []);

const items = response2.trim().match(/(\[[^\]]+\]\s*[^[]+)/g)

// 각 항목을 변수로 나누기
const results = items.map(item => {
    const title = item.match(/(\[[^\]]+\])/)[0]; // 제목 추출
    const content = item.replace(/(\[[^\]]+\]\s*)/, '').trim(
    return { title, content };
});
```

할 일: 미완성한 곰곰이의 추천 해결하기

해결법: useState로 상태를 줘서 로딩이 끝나면 후처리가 시작되도록?

해결!


```
const [results, setResults] = useState([]); //results 상태 추가

.
.
.

useEffect(() => {
  if(response2){
    const items = response2.match(/(\[[^\]]+\]\s*[^[]+)/g);

    if(items) {
      const P_results = items.map(item => {
        const title = item.match(/(\[[^\]]+\])/)[0]; // 제목 추
        const content = item.replace(/(\[[^\]]+\]\s*)/, '').t
        return {title, content};
      });
      setResults(P_results);
    }
  }
}, [response2]);
```

```
      //results에 대한 조건을 걸어서 results가 존재할 때 렌더링
      {loading2 ? (
        <p>Loading...</p>
      ) : (
```

```
        <div>
          {results && results.length === 4 ? (
            <>
              <div className={styles.book_title}>{results[0].
              <div className={styles.book_content}>{results[0
              <div className={styles.movie_title}>{results[1]
              <div className={styles.movie_content}>{results[
              <div className={styles.music_title}>{results[2]
              <div className={styles.music_content}>{results[
              <div className={styles.food_title}>{results[3].
              <div className={styles.food_content}>{results[3
            </>
          ):(
            <p>추천 결과를 불러올 수 없어 :( </p>
          )}

        </div>
      )}
```

새로운 문제

문제: loading 중일 때 'loading...'이 안 뜸

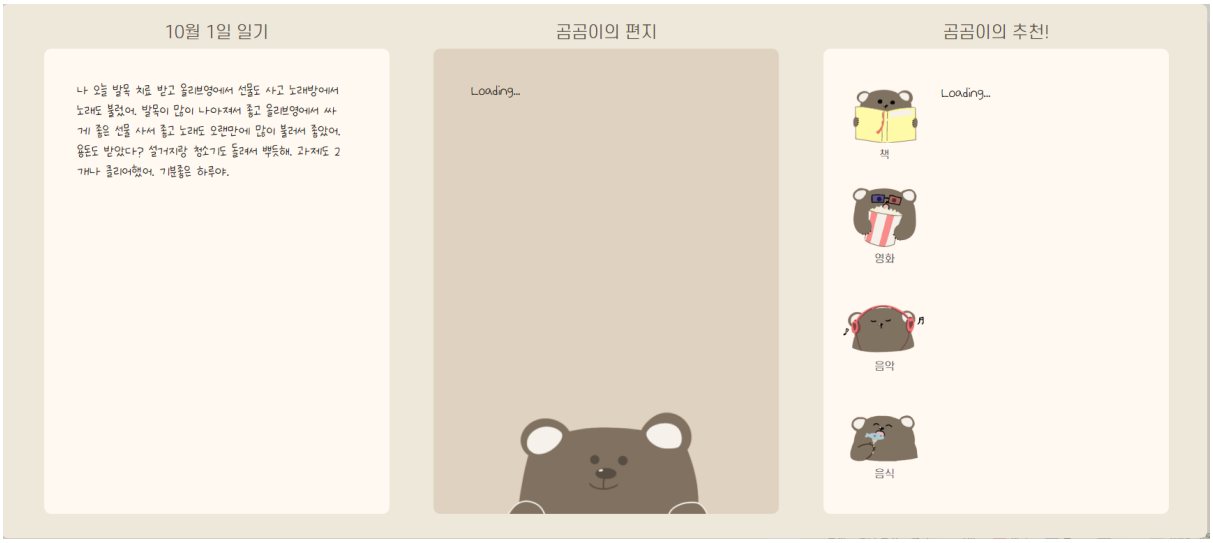
해결법: 로딩이 뜨는 조건을 변경

```
      <div>
        {!results || results.length === 0 ? (
          <div className={styles.loading}>Loading...</div>
        ) : (
          <div>

              <div className={styles.book_title}>{results[0
              <div className={styles.book_content}>{results
              <div className={styles.movie_title}>{results[
              <div className={styles.movie_content}>{result
              <div className={styles.music_title}>{results[
              <div className={styles.music_content}>{result
              <div className={styles.food_title}>{results[3
              <div className={styles.food_content}>{results
```

```
        </div>
      )}
    </div>
```

해결


10월 1일 일기
나 오늘 발목 처음 받고 올리브영에서 선물도 사고 노래방에서 노래도 불렀어. 발목이 많이 나아져서 좋고 올리브영에서 싸게 좋은 선물 사서 좋고 노래도 오랜만에 많이 불러서 좋았어. 용돈도 받았다? 설거지랑 청소기도 돌려서 뿌듯해. 과제도 2개나 끌리어했어. 기분좋은 하루야.
곰곰이의 편지
Loading...
곰곰이의 추천!
Loading...
책
영화
음악
음식