

7. AI 애플리케이션 개발



Unlock your creativity with LLMs

Flask를 활용한 웹서비스 개발

Flask

Microframework, essential components

Lightweight, potential for better performance

Manual configuration, flexible choices

Open style, gentler learning curve

Highly flexible, modular architecture

Small to medium projects, startups

■ Flask 설치

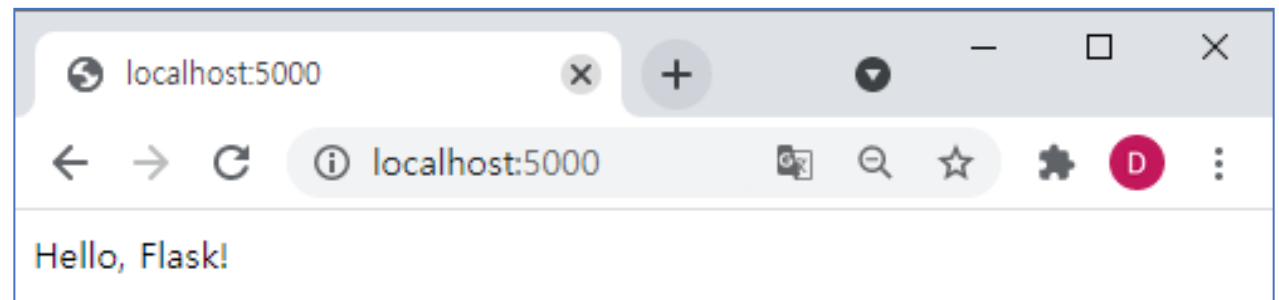
- pip install flask

■ 프로그램 작성 : app.py

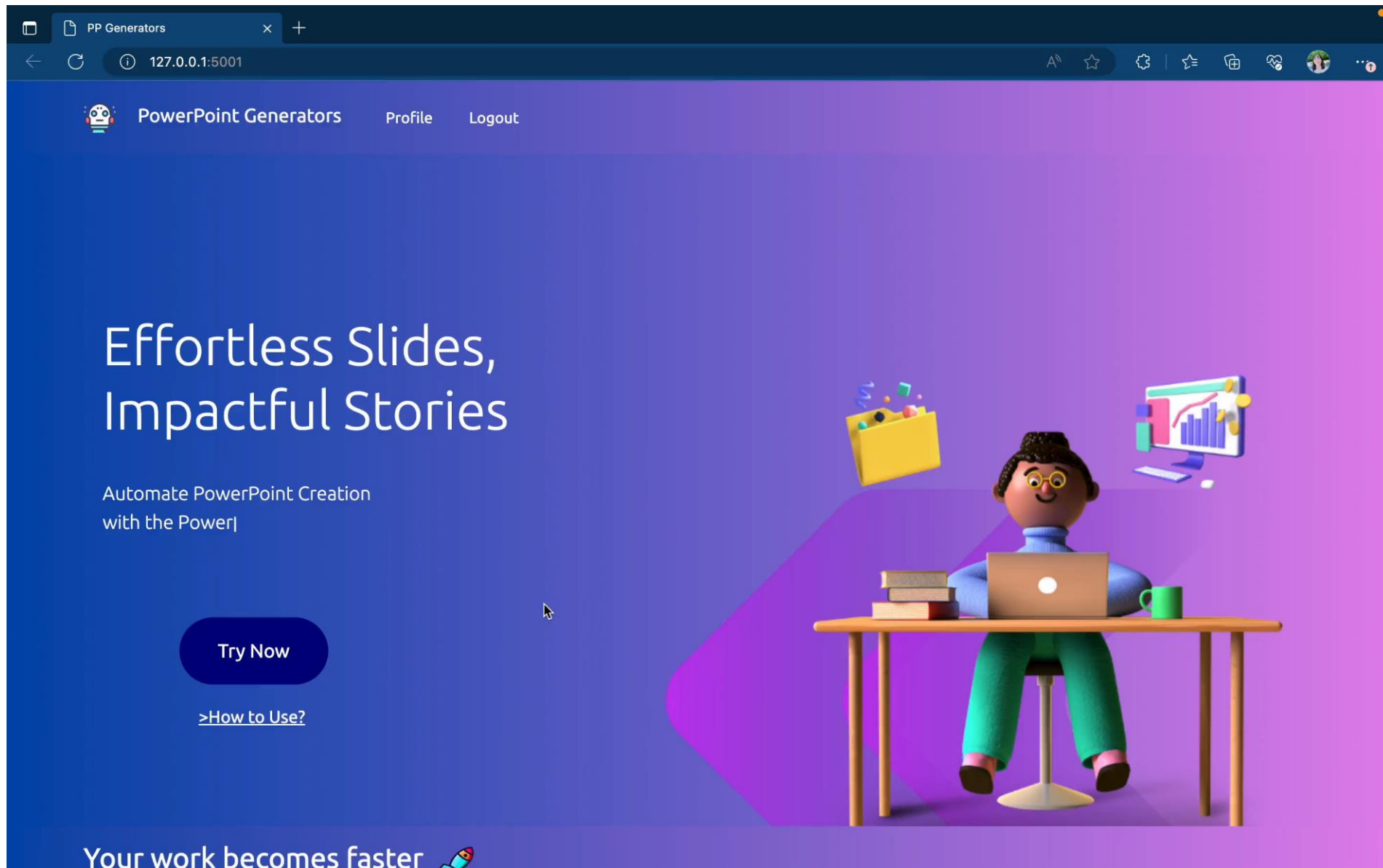
```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/index')
6 @app.route('/')
7 def hello():
8     return 'Hello, Flask!'
```

■ 프로그램 실행

- flask run
- 웹브라우저에서 <http://127.0.0.1:5000/> 접속



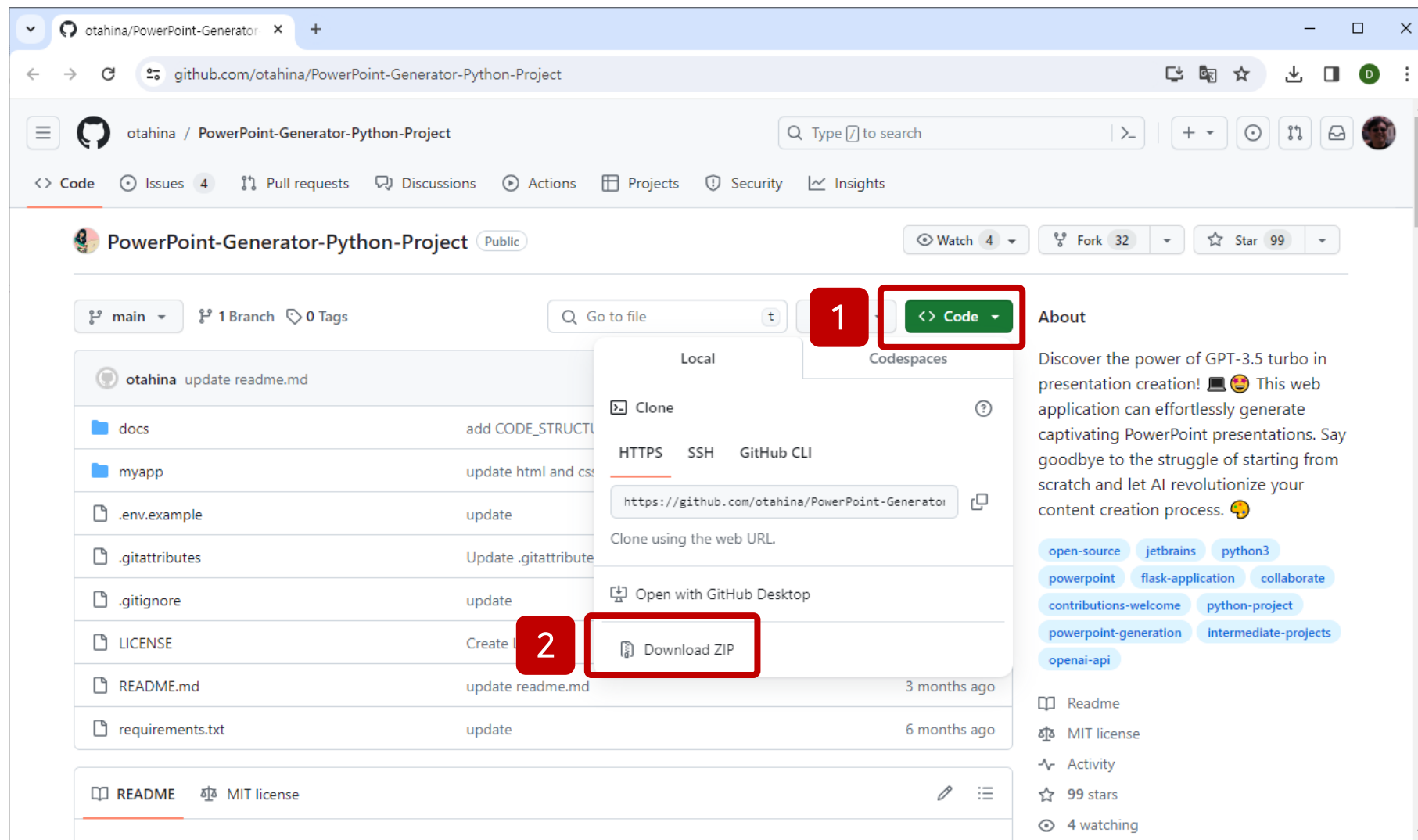
PPT 생성 웹서비스 개발



PPT 생성 웹서비스 개발

■ 소스 다운로드

<https://github.com/otahina/PowerPoint-Generator-Python-Project>



PPT 생성 웹서비스 개발

1. 소스 디렉토리에 있는 .env.example 파일명을 .env 로 변경하고 API 키를 입력

```
# Secret Key for Flask(You can decide on your own)
SECRET_KEY=your_secret_key
```

```
# OpenAI API Key
OPENAI_API_KEY=your_openai_key
```

```
# Pexels API Key
PEXELS_API_KEY=your_pexels_key
```

<https://www.pexels.com/api/>

2. 파이썬 가상환경 생성 및 패키지 설치

```
python -m venv myenv
```

```
(Windows) myenv\Scripts\activate.bat
```

```
(Linux / macOS) source myenv/bin/activate
```

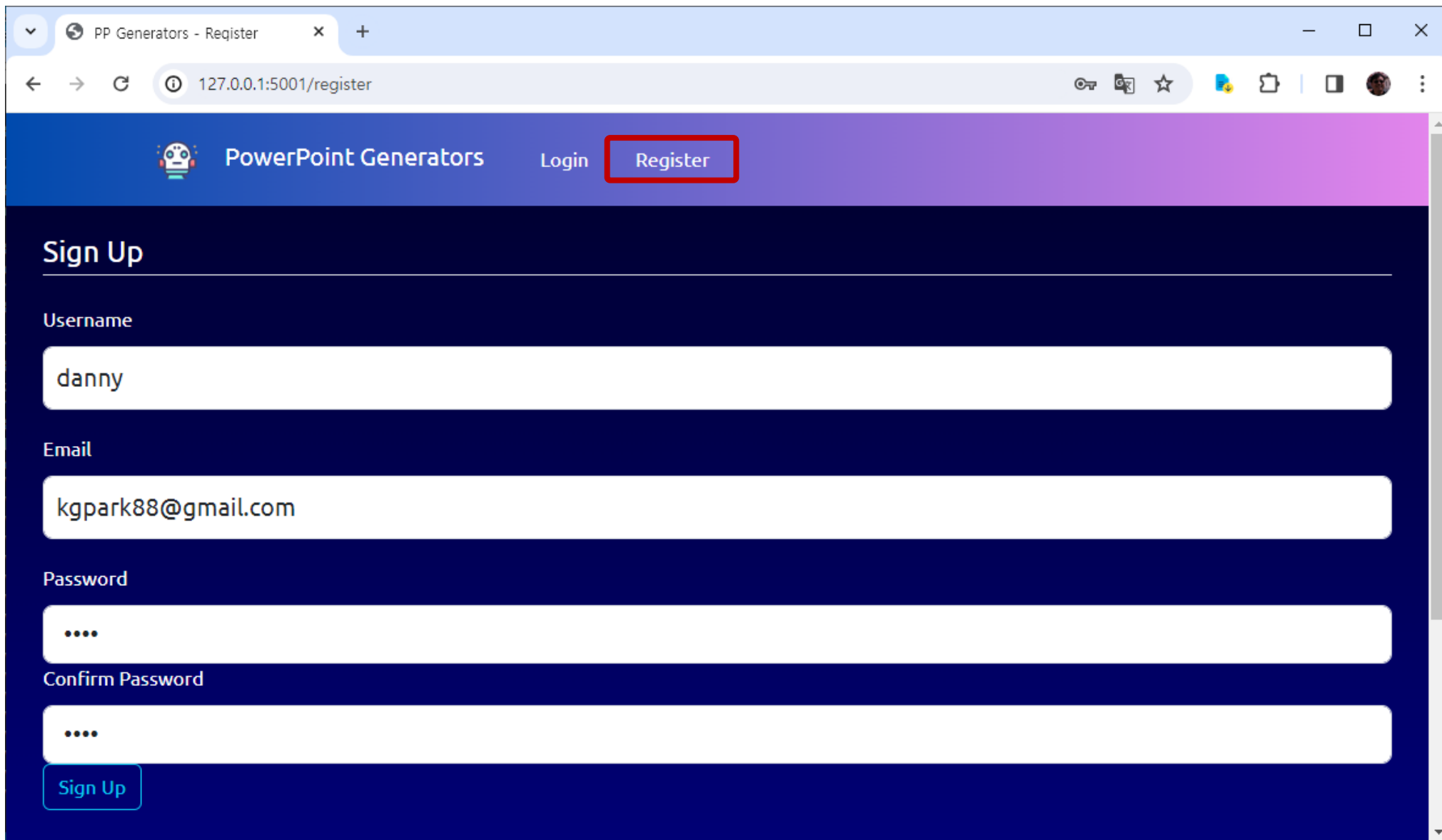
```
pip install -r requirements.txt
```

3. 프로그램 실행

```
cd myapp
```

```
python flaskapp.py
```

PPT 생성 웹서비스 - 가입 화면



The screenshot shows a web browser window with the title "PP Generators - Register". The address bar shows the URL "127.0.0.1:5001/register". The page has a dark blue header with a robot icon, the text "PowerPoint Generators", and links for "Login" and "Register". The "Register" link is highlighted with a red rectangle. Below the header, the page is titled "Sign Up" and contains four input fields: "Username" (containing "danny"), "Email" (containing "kgpark88@gmail.com"), "Password" (containing four dots), and "Confirm Password" (containing four dots). A "Sign Up" button is located at the bottom left of the form.

PP Generators - Register

127.0.0.1:5001/register

PowerPoint Generators Login Register

Sign Up

Username

danny

Email

kgpark88@gmail.com

Password

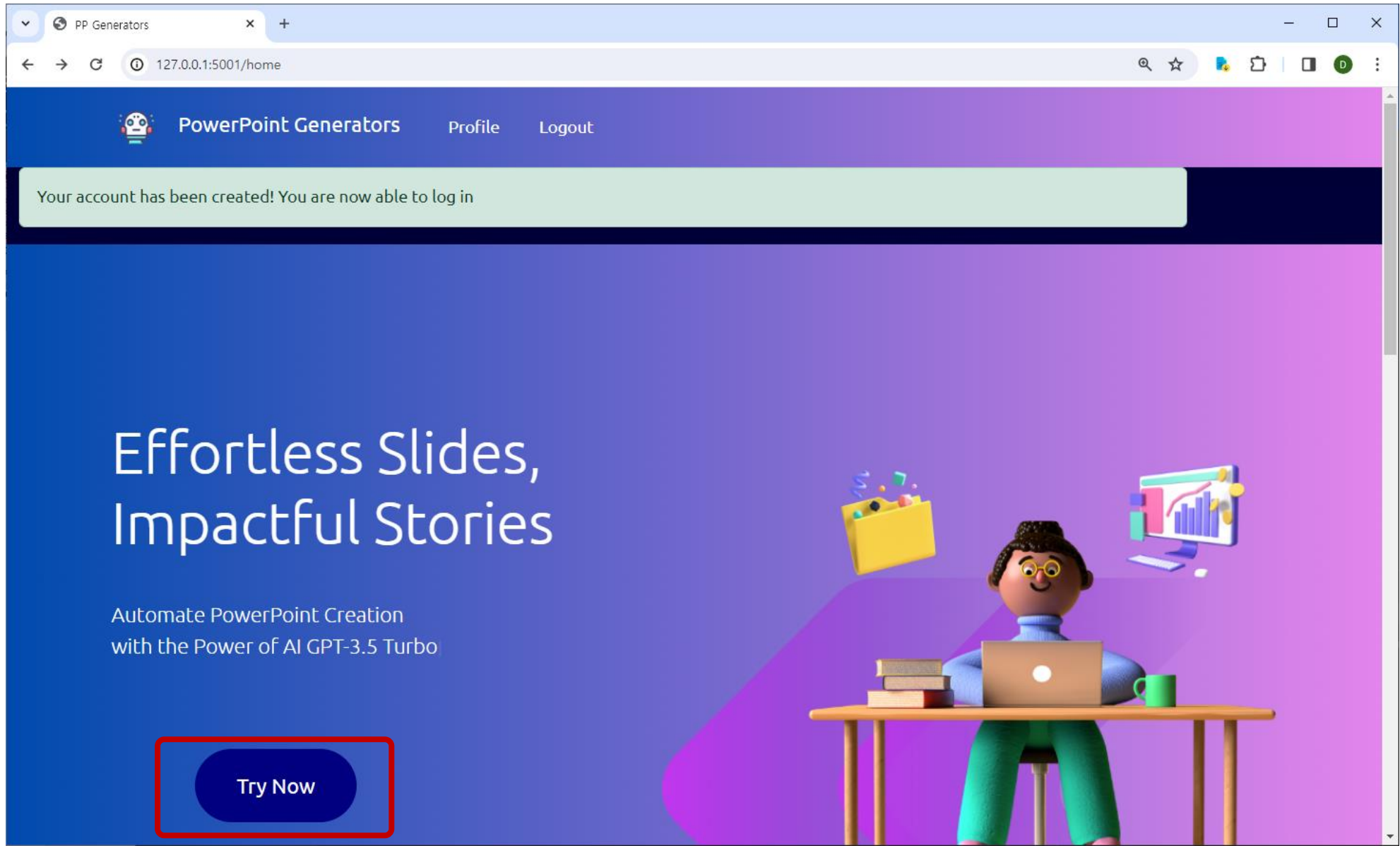
....

Confirm Password

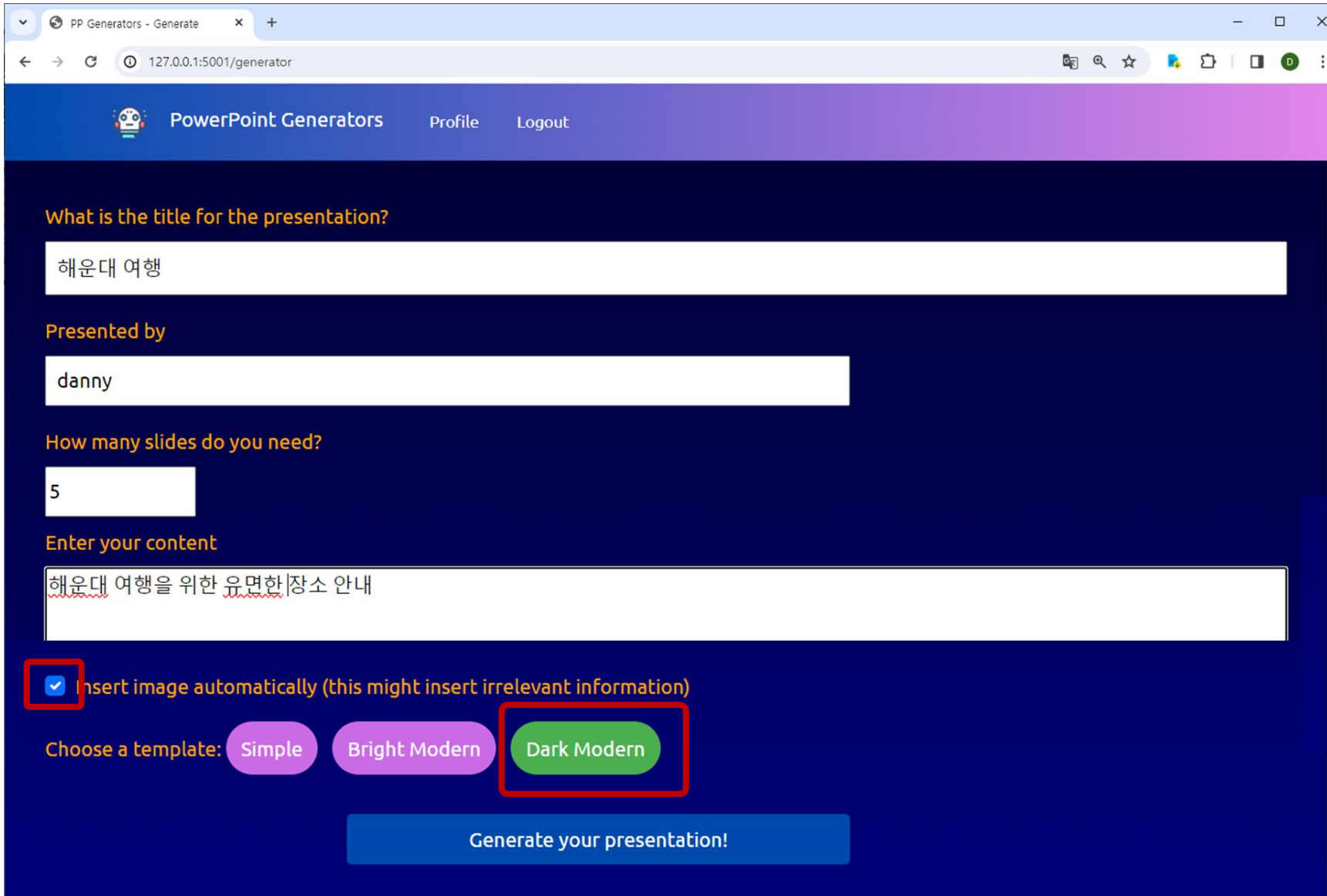
....

Sign Up

PPT 생성 웹서비스 - 메인 화면



PPT 생성 웹서비스 - 파워포인트 생성



The screenshot shows a web browser window with the URL `127.0.0.1:5001/generator`. The page has a dark blue header with the text "PowerPoint Generators" and links for "Profile" and "Logout". The main content area is dark blue and contains several input fields and a checkbox. The first input field is labeled "What is the title for the presentation?" and contains the text "해운대 여행". The second input field is labeled "Presented by" and contains the text "danny". The third input field is labeled "How many slides do you need?" and contains the text "5". The fourth input field is labeled "Enter your content" and contains the text "해운대 여행을 위한 유명한 장소 안내". Below the content field is a checkbox labeled "Insert image automatically (this might insert irrelevant information)" which is checked. At the bottom, there are three buttons for "Choose a template": "Simple", "Bright Modern", and "Dark Modern". The "Dark Modern" button is highlighted with a red box. A large blue button at the bottom center says "Generate your presentation!".

PP Generators - Generate x +

127.0.0.1:5001/generator

PowerPoint Generators Profile Logout

What is the title for the presentation?

해운대 여행

Presented by

danny

How many slides do you need?

5

Enter your content

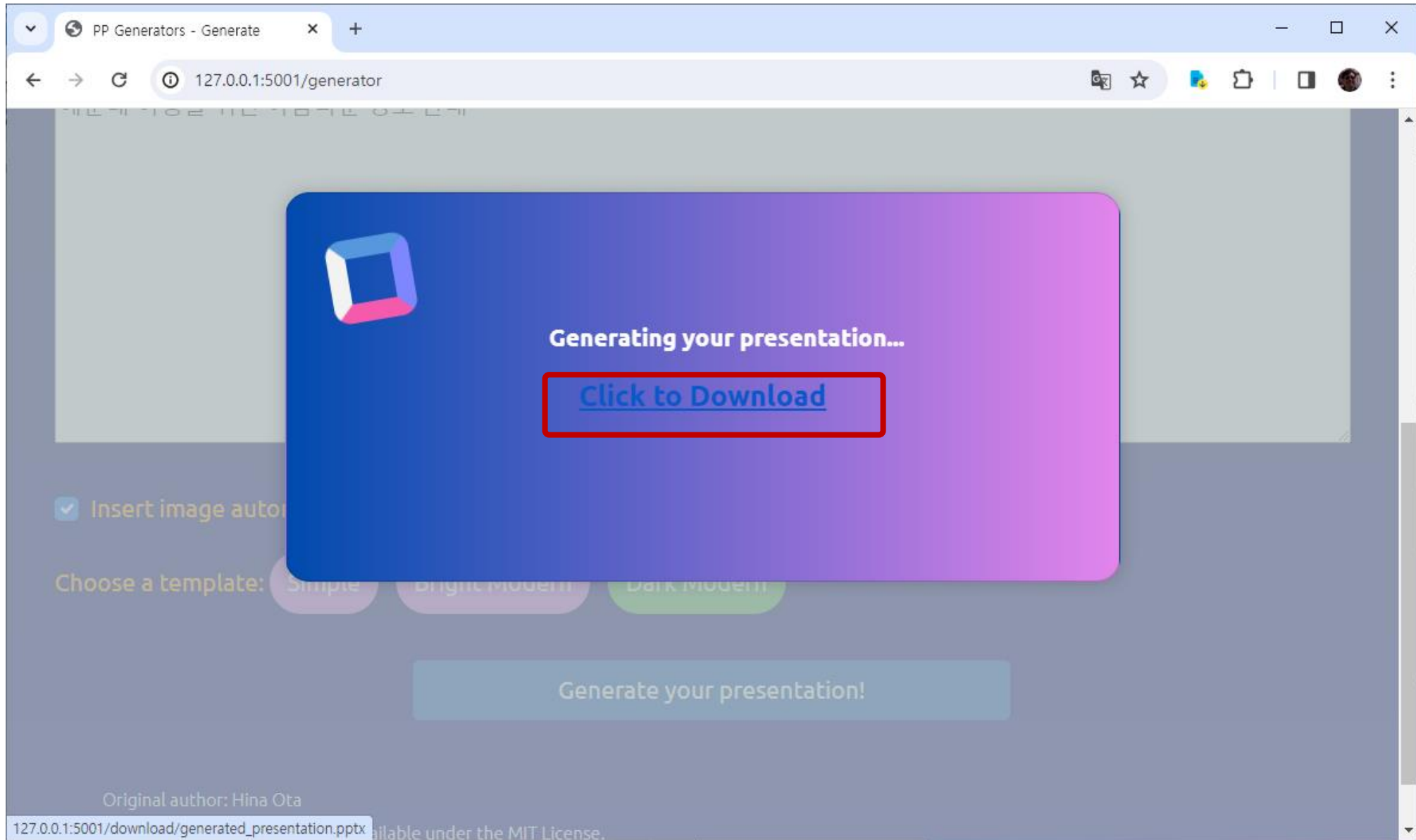
해운대 여행을 위한 유명한 장소 안내

☒ Insert image automatically (this might insert irrelevant information)

Choose a template: Simple Bright Modern Dark Modern

Generate your presentation!

PPT 생성 웹서비스 - PPT 다운로드



PPT 생성 웹서비스 - 생성된 PPT

해운대 여행

Presented by danny

1

Introduction

- - Introduction to Haeundae, a popular travel destination in Busan, South Korea
- - Overview of the presentation agenda and main attractions to be covered
- Keyword: Haeundae Travel



2

Haeundae Beach

- - Description of Haeundae Beach and its significance as a major tourist attraction
- - Highlighting the beautiful sandy beach, clear blue water, and vibrant atmosphere
- - Mentioning various water sports and activities available
- Keyword: Haeundae Beach



3

Dongbaek Island

- - Introduction to Dongbaek Island, a scenic spot located near Haeundae Beach
- - Mentioning the picturesque walking trails, lush greenery, and stunning ocean views
- - Noting the historical significance of the island and its local legends
- Keyword: Dongbaek Island



Haeundae Traditional Market

- - Showcasing Haeundae Traditional Market, a bustling market known for its local produce and street food
- - Describing the vibrant atmosphere, diverse range of food options, and unique shopping experience
- - Highlighting popular food items, such as Korean snacks
- Keyword: Traditional Market



Haedong Yonggungsa Temple

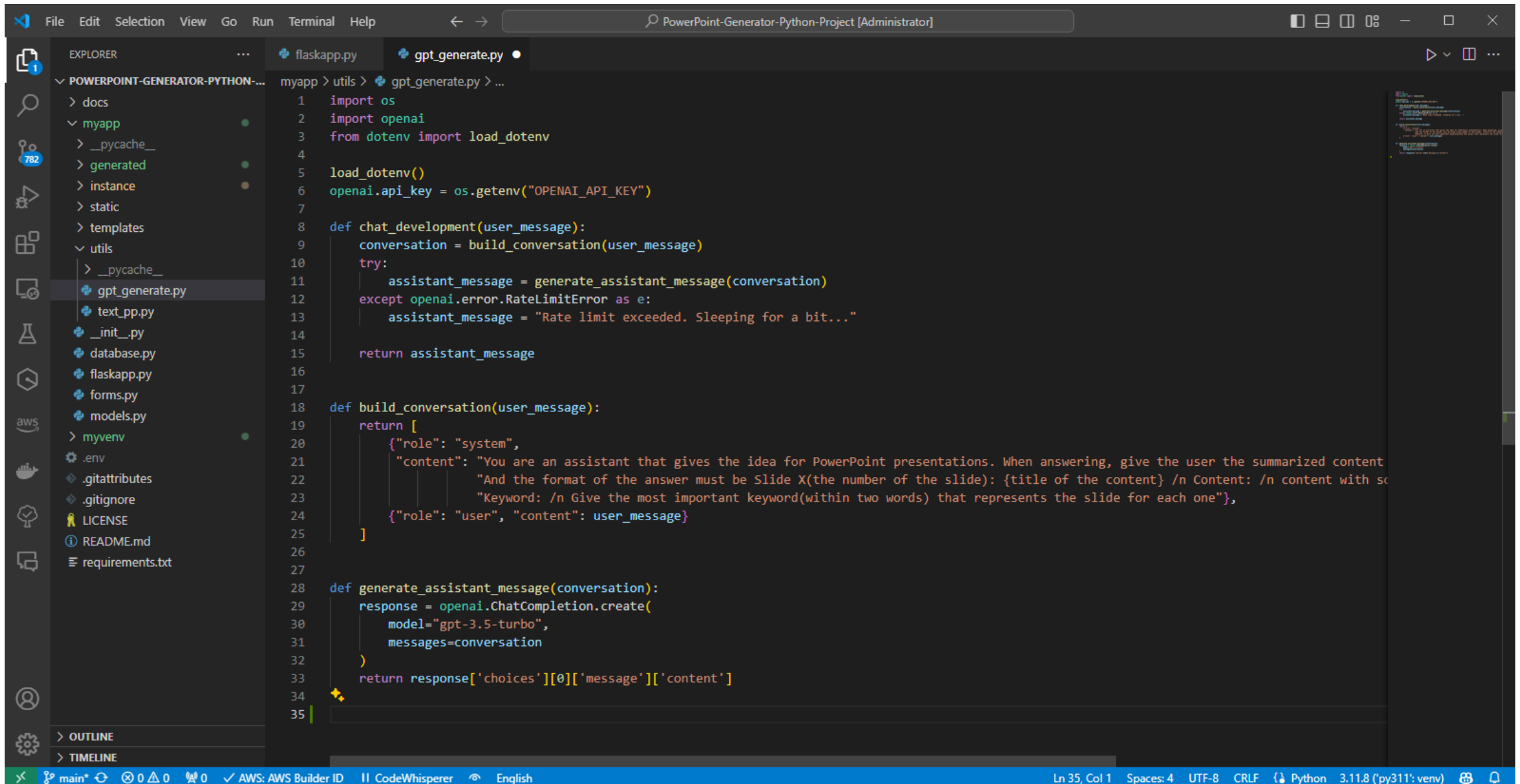
- - Presenting Haedong Yonggungsa Temple, a famous seaside Buddhist temple in Haeundae
- - Describing the temple's stunning oceanfront location and its tranquil and serene ambiance
- - Mentioning notable features and attractions within the temple complex, such as the main hall and the pagoda
- Keyword: Yonggungsa Temple



실습 - 코드 분석 : flaskapp.py

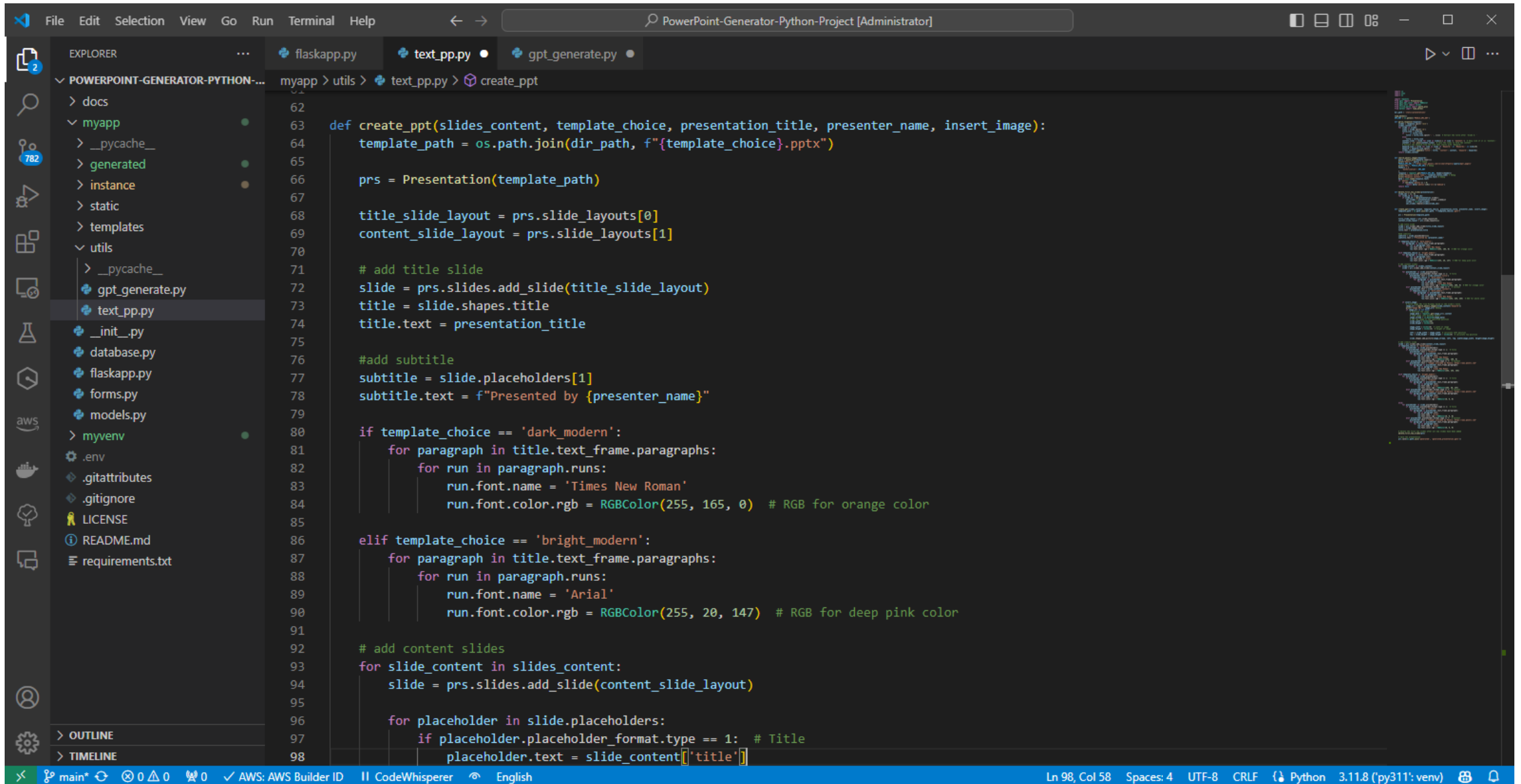
```
83
84 @app.route('/generator', methods=['GET', 'POST'])
85 def generate():
86     if request.method == 'POST':
87         number_of_slide = request.form.get('number_of_slide')
88         user_text = request.form.get('user_text')
89         template_choice = request.form.get('template_choice')
90         presentation_title = request.form.get('presentation_title')
91         presenter_name = request.form.get('presenter_name')
92         insert_image = 'insert_image' in request.form
93
94         user_message = f"I want you to come up with the idea for the PowerPoint. The number of slides is {number_of_slide}. " \
95             f"The content is: {user_text}. The title of content for each slide must be unique, " \
96             f"and extract the most important keyword within two words for each slide. Summarize the content for each slide. "
97
98         assistant_response = chat_development(user_message)
99         # Check the response (for debug)
100         print(f"Assistant Response:\n{assistant_response}")
101         slides_content = parse_response(assistant_response)
102         create_ppt(slides_content, template_choice, presentation_title, presenter_name, insert_image)
103
104     return render_template('generator.html', title='Generate')
105
106
107 @app.route('/download/<filename>', methods=['GET'])
108 def download_file(filename):
109     try:
110         return send_from_directory('generated', filename, as_attachment=True)
111
112     except FileNotFoundError:
113         abort(404)
114
115
116 if __name__ == "__main__":
117     with app.app_context():
118         db.create_all()
119     app.run(port=5001, debug=True)
```

실습 - 코드 분석 : gpt_generate.py



```
1 import os
2 import openai
3 from dotenv import load_dotenv
4
5 load_dotenv()
6 openai.api_key = os.getenv("OPENAI_API_KEY")
7
8 def chat_development(user_message):
9     conversation = build_conversation(user_message)
10    try:
11        assistant_message = generate_assistant_message(conversation)
12    except openai.error.RateLimitError as e:
13        assistant_message = "Rate limit exceeded. Sleeping for a bit..."
14
15    return assistant_message
16
17
18 def build_conversation(user_message):
19     return [
20         {"role": "system",
21          "content": "You are an assistant that gives the idea for PowerPoint presentations. When answering, give the user the summarized content\n\nAnd the format of the answer must be Slide X(the number of the slide): {title of the content} /n Content: /n content with s\n\nKeyword: /n Give the most important keyword(within two words) that represents the slide for each one"},
22         {"role": "user", "content": user_message}
23     ]
24
25
26
27
28 def generate_assistant_message(conversation):
29     response = openai.ChatCompletion.create(
30         model="gpt-3.5-turbo",
31         messages=conversation
32     )
33     return response['choices'][0]['message']['content']
34
35
```

실습 - 코드 분석 : text_pp.py



```
62
63 def create_ppt(slides_content, template_choice, presentation_title, presenter_name, insert_image):
64     template_path = os.path.join(dir_path, f"{template_choice}.pptx")
65
66     prs = Presentation(template_path)
67
68     title_slide_layout = prs.slide_layouts[0]
69     content_slide_layout = prs.slide_layouts[1]
70
71     # add title slide
72     slide = prs.slides.add_slide(title_slide_layout)
73     title = slide.shapes.title
74     title.text = presentation_title
75
76     #add subtitle
77     subtitle = slide.placeholders[1]
78     subtitle.text = f"Presented by {presenter_name}"
79
80     if template_choice == 'dark_modern':
81         for paragraph in title.text_frame.paragraphs:
82             for run in paragraph.runs:
83                 run.font.name = 'Times New Roman'
84                 run.font.color.rgb = RGBColor(255, 165, 0) # RGB for orange color
85
86     elif template_choice == 'bright_modern':
87         for paragraph in title.text_frame.paragraphs:
88             for run in paragraph.runs:
89                 run.font.name = 'Arial'
90                 run.font.color.rgb = RGBColor(255, 20, 147) # RGB for deep pink color
91
92     # add content slides
93     for slide_content in slides_content:
94         slide = prs.slides.add_slide(content_slide_layout)
95
96         for placeholder in slide.placeholders:
97             if placeholder.placeholder_format.type == 1: # Title
98                 placeholder.text = slide_content['title']]
```


실습 - 기획/제안서 생성AI앱 개발



auto-generator.ipynb



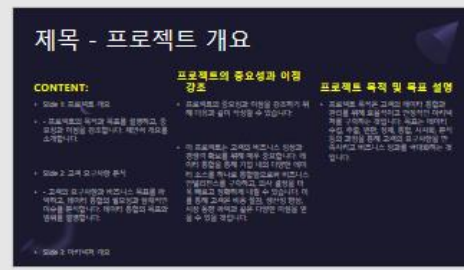
1



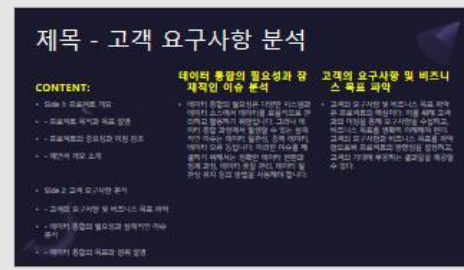
2



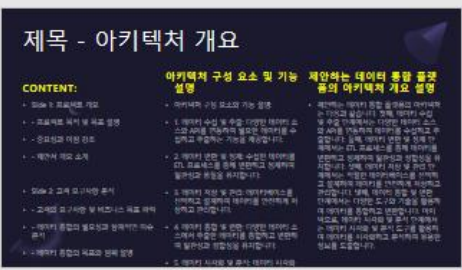
3



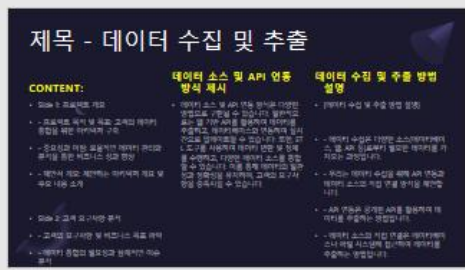
4



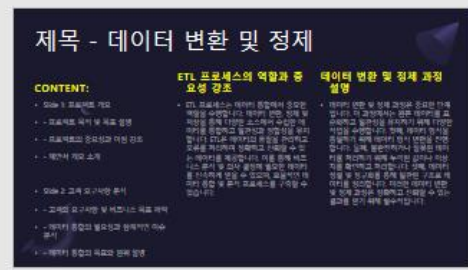
5



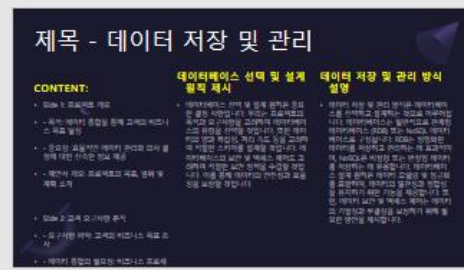
6



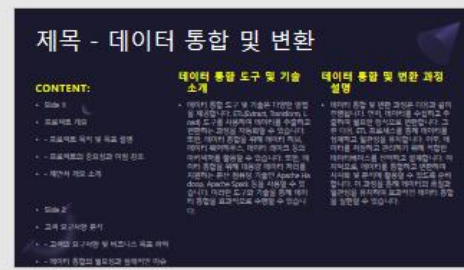
7



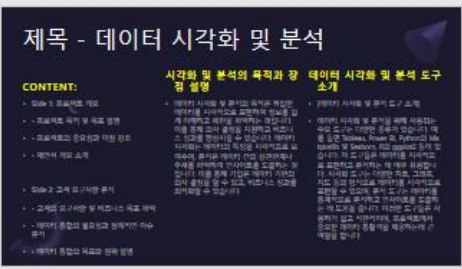
8



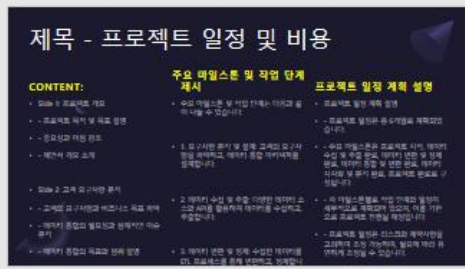
9



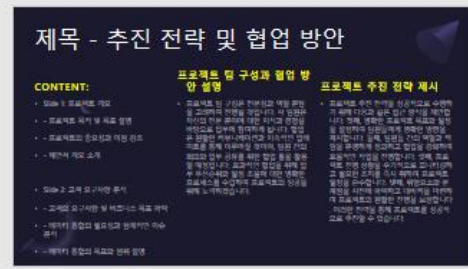
10



11



12



13



14

AI로 만들기

어떻게 시작하시겠어요?



텍스트로 붙여넣기
노트, 개요 또는 기존 콘텐츠에서 만들기

계속 →



생성
몇 초 만에 한 줄 프롬프트에서 만들기

→




파일 가져오기
기존 문서와 프레젠테이션을 변환하거나 개선


→

생성


오늘은 무엇을 만들고 싶으신가요?



프레젠테이션



문서



웹 페이지

8 카드 ▾ 한국어 ▾

데이터 통합 플랫폼 구축

개요 생성 →

gamma.app/create/generate x +

gamma.app/create/generate/imi10ieywh4thvh

← 뒤로 생성

프롬프트 8 카드 ▾ 한국어 ▾

데이터 통합 플랫폼 구축

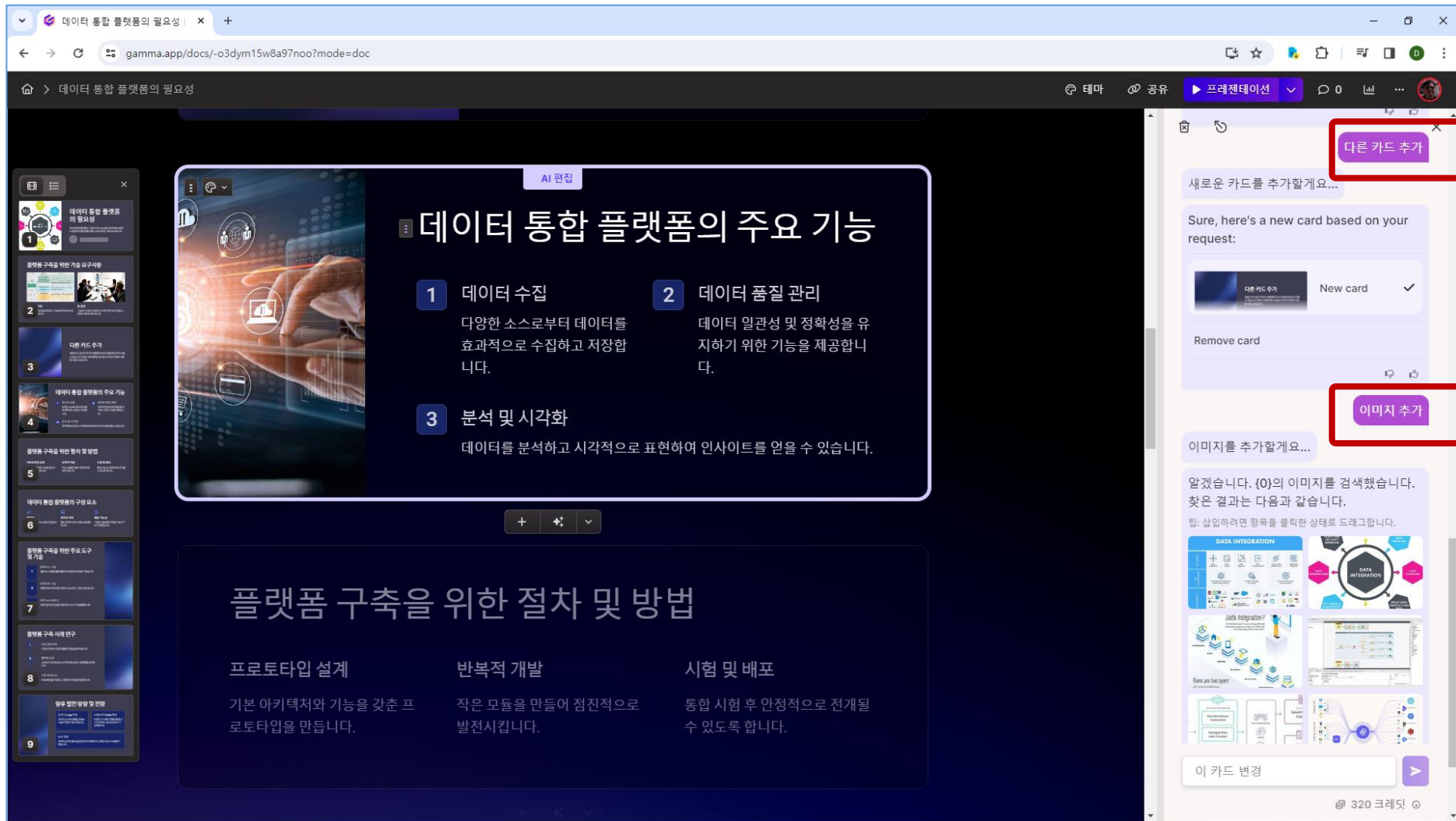
윤곽선

- 1 데이터 통합 플랫폼의 필요성
- 2 플랫폼 구축을 위한 기술 요구사항
- 3 데이터 통합 플랫폼의 주요 기능
- 4 플랫폼 구축을 위한 절차 및 방법
- 5 데이터 통합 플랫폼의 구성 요소
- 6 플랫폼 구축을 위한 주요 도구 및 기술
- 7 플랫폼 구축 사례 연구
- 8 향후 발전 방향 및 전망

+ 카드 추가

400 크레딧 8 총 카드 수

계속 40 → ?



데이터 통합 플랫폼의 필요성

데이터 통합 플랫폼의 주요 기능

- 1 데이터 수집**
다양한 소스로부터 데이터를 효과적으로 수집하고 저장합니다.
- 2 데이터 품질 관리**
데이터 일관성 및 정확성을 유지하기 위한 기능을 제공합니다.
- 3 분석 및 시각화**
데이터를 분석하고 시각적으로 표현하여 인사이트를 얻을 수 있습니다.

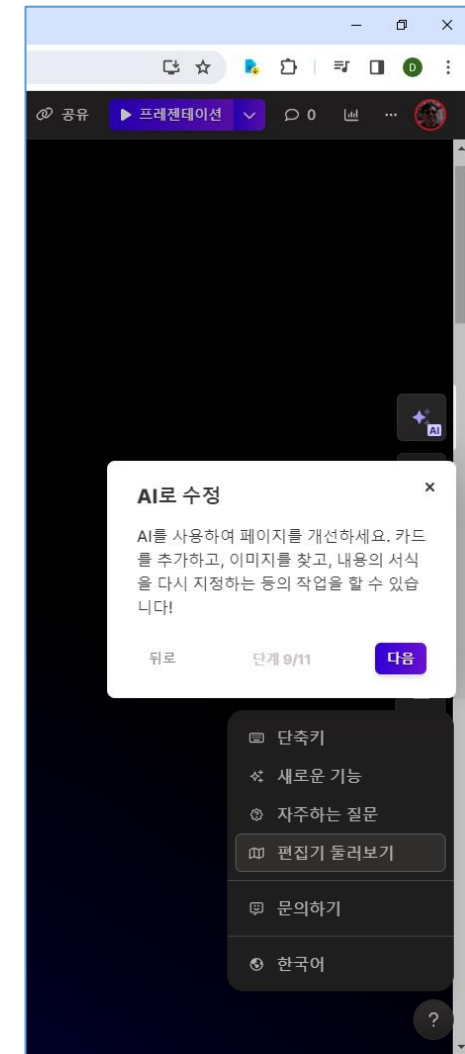
플랫폼 구축을 위한 절차 및 방법

프로토타입 설계
기본 아키텍처와 기능을 갖춘 프로토타입을 만듭니다.

반복적 개발
작은 모듈을 만들어 점진적으로 발전시킵니다.

시험 및 배포
통합 시험 후 안정적으로 전개될 수 있도록 합니다.

320 크레딧



데이터 통합 플랫폼의 필요성

데이터 통합 플랫폼의 주요 기능

- 1 데이터 수집**
다양한 소스로부터 데이터를 효과적으로 수집하고 저장합니다.
- 2 데이터 품질 관리**
데이터 일관성 및 정확성을 유지하기 위한 기능을 제공합니다.
- 3 분석 및 시각화**
데이터를 분석하고 시각적으로 표현하여 인사이트를 얻을 수 있습니다.

플랫폼 구축을 위한 절차 및 방법

프로토타입 설계
기본 아키텍처와 기능을 갖춘 프로토타입을 만듭니다.

반복적 개발
작은 모듈을 만들어 점진적으로 발전시킵니다.

시험 및 배포
통합 시험 후 안정적으로 전개될 수 있도록 합니다.

320 크레딧

실습 - 유튜브 요약



YouTube Loader.ipynb

실습 - LangServe

LangServe는 FastAPI와 통합되어 있으며 LangChain Runnable과 Chain을 REST API로 쉽게 배포할 수 있습니다.



- 입력 및 출력 자동 추론
- 오류 메시지 표시
- 엔드포인트 제공

/invoke

/batch

/stream

/stream_log

/playground

- LangSmith 추적 기능 내장

<https://python.langchain.com/docs/langserve>



FastAPI framework, high performance, easy to learn, fast to code, ready for production

- **Fast:** Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic).
One of the fastest Python frameworks available.
- **Fast to code:** Increase the speed to develop features by about 200% to 300%. *
- **Fewer bugs:** Reduce about 40% of human (developer) induced errors. *
- **Intuitive:** Great editor support. Completion everywhere. Less time debugging.
- **Easy:** Designed to be easy to use and learn. Less time reading docs.
- **Short:** Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.
- **Robust:** Get production-ready code. With automatic interactive documentation.
- **Standards-based:** Based on (and fully compatible with) the open standards for APIs:
OpenAPI [↔] (previously known as Swagger) and JSON Schema [↔].

<https://fastapi.tiangolo.com/>

실습 - LangServe

1. Python 3.11.8 버전 설치

설치 파일 다운로드 : <https://www.python.org/downloads/>

2. Python 가상환경 설치

```
python -m venv venv
```

```
venv\Scripts\activate
```

3. LangChain 패키지 설치

```
pip install langchain langchain-openai langchain-cli
```

```
pip list
```

Package	Version
-----	-----
fastapi	0.109.2
langchain	0.1.7
langchain-cli	0.0.21
langchain-community	0.0.20
langchain-core	0.1.23
langchain-openai	0.0.6
langserve	0.0.41
langsmith	0.0.87
openai	1.12.0
pydantic	2.6.1
pydantic_core	2.16.2

4. 환경변수에 API KEY 추가

Windows

```
setx OPENAI_API_KEY=sk-...
```

```
setx LANGCHAIN_TRACING_V2=true
```

```
setx LANGCHAIN_API_KEY=<your-api-key>
```

```
setx LANGCHAIN_PROJECT=<your-project>
```

macOS/Linux

```
export OPENAI_API_KEY=sk-...
```

```
export LANGCHAIN_TRACING_V2=true
```

```
export LANGCHAIN_API_KEY=<your-api-key>
```

```
export LANGCHAIN_PROJECT=<your-project>
```

5. LangChain 프로젝트 생성

```
langchain app new my-app
```

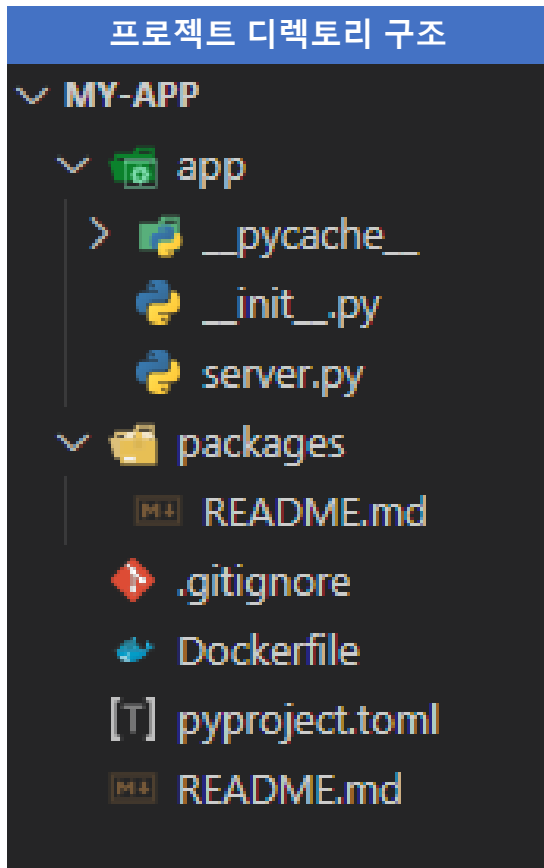
실습 - LangServe



my-app

6. 프로그램 개발

my-app/app/server.py 파일 수정



```
from fastapi import FastAPI
from fastapi.responses import RedirectResponse
from langserve import add_routes
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate

app = FastAPI()

@app.get("/")
async def redirect_root_to_docs():
    return RedirectResponse("/docs")

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "사용자 입력을 해적 말투로 번역",
        ),
        ("human", "{text}"),
    ]
)
model = ChatOpenAI()
chain = prompt | model

add_routes(app, chain, path="/pirate-speak")

if __name__ == "__main__":
    import uvicorn

    uvicorn.run(app, host="0.0.0.0", port=8000)
```

실습 - LangServe



my-app

7. 프로그램 실행

cd my-app

langchain serve

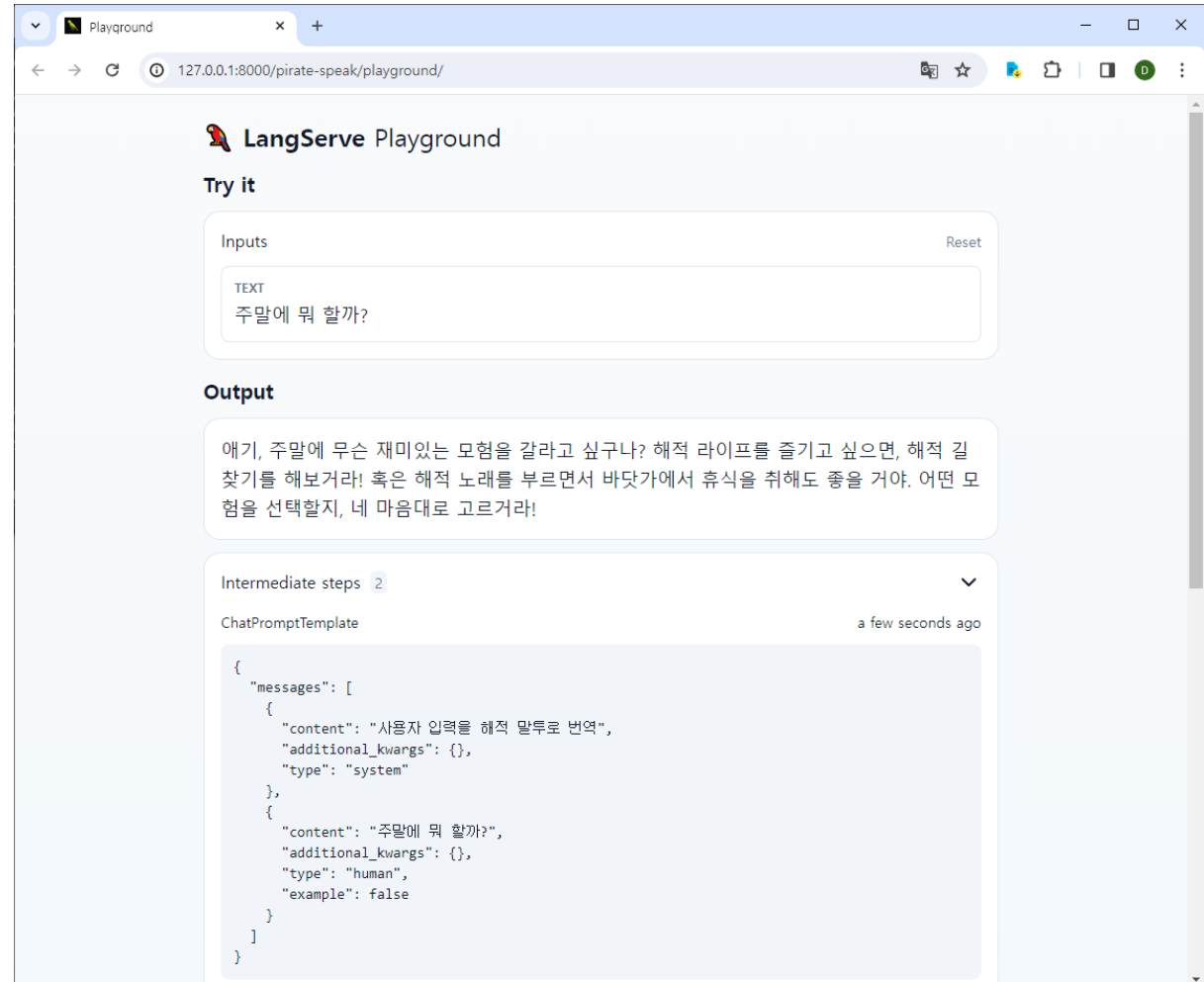
```
관리자: C:\Windows\System32\cmd.exe - langchain serve
[32mINFO-[0m:   Uvicorn running on *([1mhttp://127.0.0.1:8000-[0m (Press CTRL+C to quit)
[32mINFO-[0m:   Started reloader process [*([36m[1m3528-[0m] using [*([36m[1mStatReload-[0m
[32mINFO-[0m:   Started server process [*([36m[22108-[0m]
[32mINFO-[0m:   Waiting for application startup.

LANGSERVE

[1;32;40mLANGSERVE::[0m Playground for chain "/pirate-speak/" is live at:
[1;32;40mLANGSERVE::[0m |
[1;32;40mLANGSERVE::[0m |  -> /pirate-speak/playground/
[1;32;40mLANGSERVE::[0m |
[1;32;40mLANGSERVE::[0m | See all available routes at /docs/
[1;31;40mLANGSERVE::[0m ⚠ Using pydantic 2.6.1. OpenAPI docs for invoke, batch, stream, stream_log endpoints will not
be generated. API endpoints and playground should work as expected. If you need to see the docs, you can downgrade to py
dantic 1. For example, 'pip install pydantic==1.10.13'. See https://github.com/tiangolo/fastapi/issues/10360 for details
[32mINFO-[0m:   Application startup complete.
```

8. Playground

<http://127.0.0.1:8000/pirate-speak/playground/>



실습 - LangServe Examples

























설명	링크
LLMs Minimal example that reserves OpenAI and Anthropic chat models. Uses async, supports batching and streaming.	server , client
Retriever Simple server that exposes a retriever as a runnable.	server , client
Conversational Retriever A Conversational Retriever exposed via LangServe	server , client
Agent without conversation history based on OpenAI tools	server , client
Agent with conversation history based on OpenAI tools	server , client
RunnableWithMessageHistory to implement chat persisted on backend, keyed off a session_id supplied by client.	server , client
RunnableWithMessageHistory to implement chat persisted on backend, keyed off a conversation_id supplied by client, and user_id (see Auth for implementing user_id properly).	server , client
Configurable Runnable to create a retriever that supports run time configuration of the index name.	server , client
Configurable Runnable that shows configurable fields and configurable alternatives.	server , client
APIHandler Shows how to use APIHandler instead of add_routes. This provides more flexibility for developers to define endpoints. Works well with all FastAPI patterns, but takes a bit more effort.	server
LCEL Example Example that uses LCEL to manipulate a dictionary input.	server , client
Auth with add_routes: Simple authentication that can be applied across all endpoints associated with app. (Not useful on its own for implementing per user logic.)	server
Auth with add_routes: Simple authentication mechanism based on path dependencies. (Not useful on its own for implementing per user logic.)	server
Auth with add_routes: Implement per user logic and auth for endpoints that use per request config modifier. (Note: At the moment, does not integrate with OpenAPI docs.)	server , client
Auth with APIHandler: Implement per user logic and auth that shows how to search only within user owned documents.	server , client
Widgets Different widgets that can be used with playground (file upload and chat)	server
Widgets File upload widget used for LangServe playground.	server , client



































Awesome LangChain

Open Source Projects

Knowledge Management

- [Quiver](#): Dump your brain into your GenerativeAI Vault  Stars  28k
- [DocsGPT](#): GPT-powered chat for documentation search & assistance.  Stars  14k
- [Chaindesk](#): The no-code platform for semantic search and documents retrieval  Stars  2.8k
- [Knowledge GPT](#): Accurate answers and instant citations for your documents.  Stars  1.5k
- [Knowledge](#): Knowledge is a tool for saving, searching, accessing, and exploring all of your favorite documents and files.  Stars  1.2k
- [Anything LLM](#): A full-stack application that turns any documents into an intelligent chatbot with a easier way to manage your workspaces.  Stars  8.1k
- [DocNavigator](#): AI-powered chatbot builder that is designed to improve the user experience on pr documentation/support websites  Stars  104
- [ChatFiles](#): Upload your document and then chat with it. Powered by GPT / Embedding / TS / Next
- [DataChad](#): A streamlit app that lets you chat with any data source. Supporting both OpenAI and L GPT4All.  Stars  292
- [Second Brain AI Agent](#): A streamlit app dialog with your second brain notes using OpenAI and Ch  Stars  120
- [examor](#): A website application that allows you to take exams based on your knowledge notes. Let remember what you have learned and written.  Stars  989
- [RepoChat](#): Chatbot assistant enabling GitHub repository interaction using LLMs with Retrieval Aug Generation  Stars  194
- [SolidGPT](#): Chat everything with your code repository, ask repository level code questions, and dis requirements  Stars  1.8k

Other / Chatbots

- [DB GPT](#): Interact your data and environment using the local GPT, no data leaks, 100% privately, 100% security  Stars  9.8k
- [AudioGPT](#): Understanding and Generating Speech, Music, Sound, and Talking Head  Stars  9.6k
- [Paper QA](#): LLM Chain for answering questions from documents with citations  Stars  3.5k
- [Chat Langchain](#): locally hosted chatbot specifically focused on question answering over the LangChain documentation  Stars  4.4k
- [Langchain Chat](#): another Next.js frontend for LangChain Chat.  Stars  935
- [Book GPT](#): drop a book, start asking question.  Stars  431
- [Chat LangchainJS](#): NextJS version of Chat Langchain  Stars  286
- [Doc Search](#): converse with book - Built with GPT-3  Stars  600
- [Fact Checker](#): fact-checking LLM outputs with langchain  Stars  252
- [MM ReAct](#): Multi Modal ReAct Design
- [QABot](#): Query local or remote files or databases with natural language queries powered by langchain and openai  Stars  230
- [GPT Automator](#): Your voice-controlled Mac assistant.  Stars  191
- [Teams LangchainJS](#): Demonstration of LangChainJS with Teams / Bot Framework bots  Stars  51
- [ChatGPT](#): ChatGPT & langchain example for node.js & Docker  Stars  56
- [FlowGPT](#): Generate diagram with AI  Stars  217
- [langchain-text-summarizer](#): A sample streamlit application summarizing text using LangChain  Stars  14
- [Langchain Chat Websocket](#): About LangChain LLM chat with streaming response over websockets  Stars  71



Gradio Tools

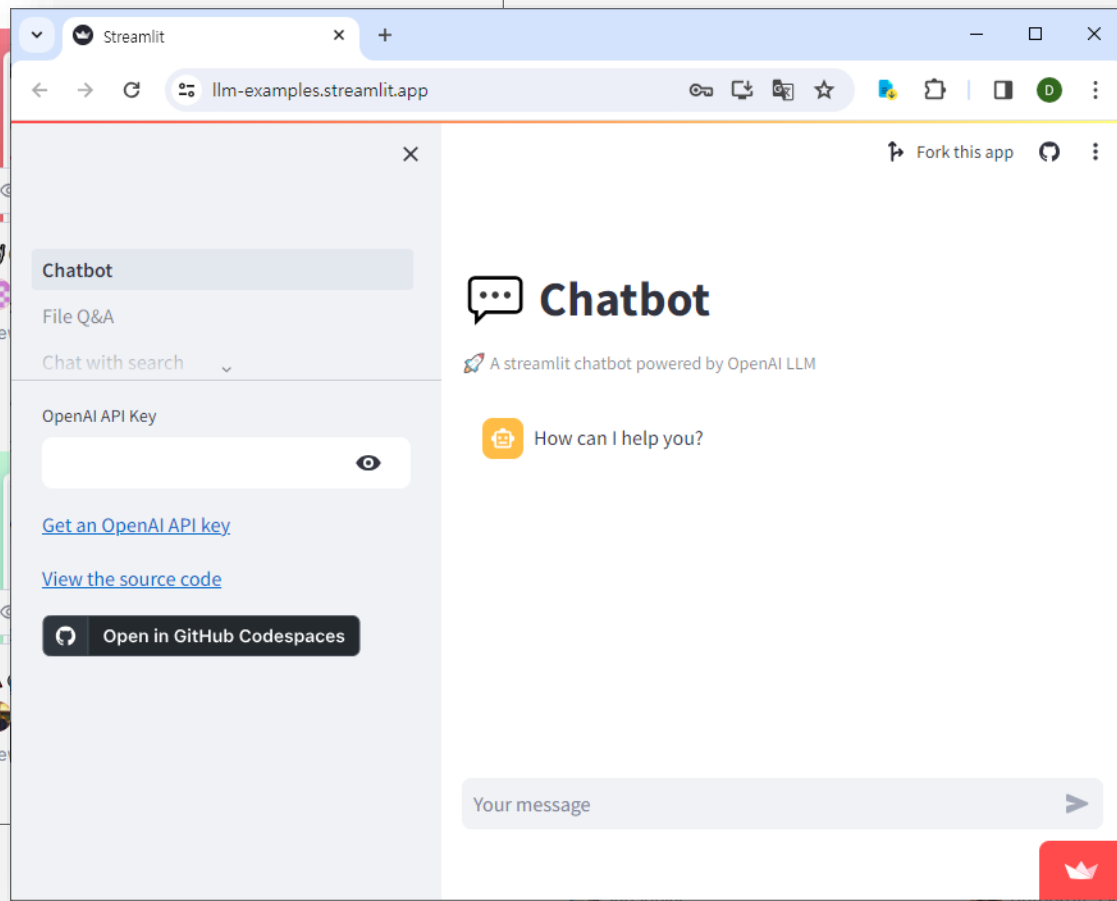
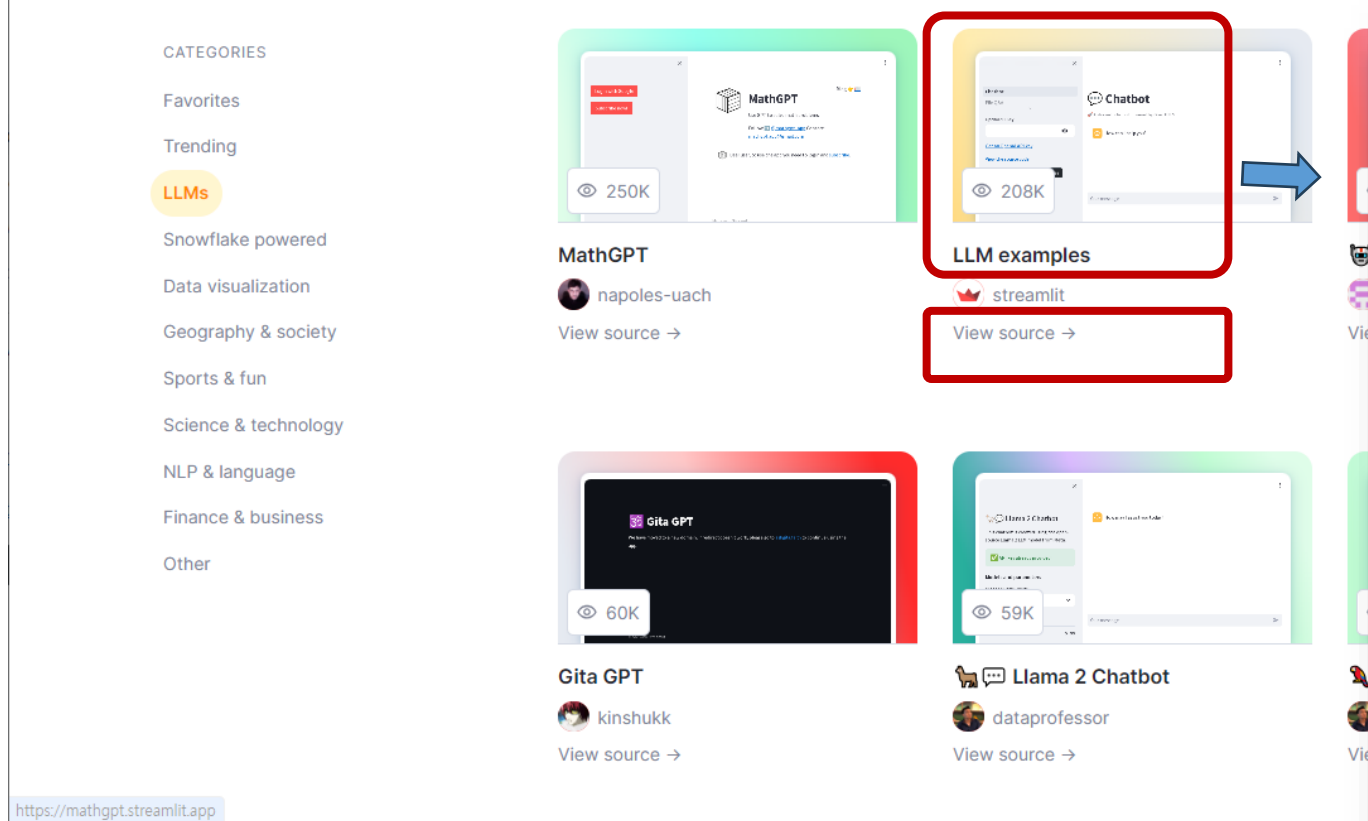
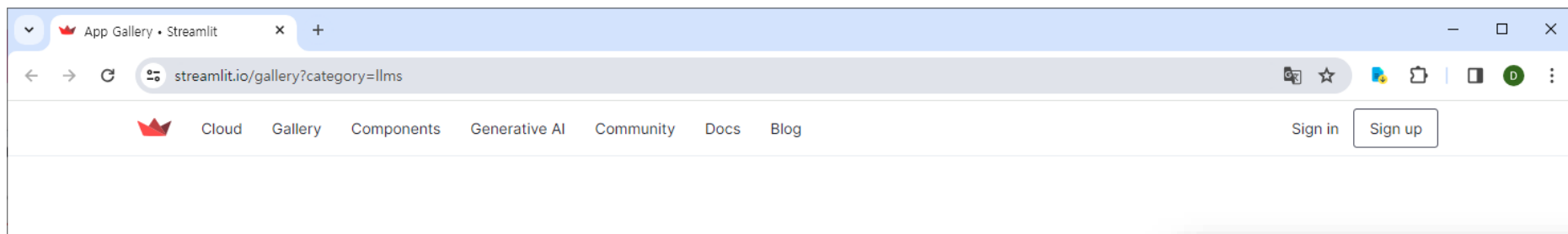
Gradio  LLM Agents




Streamlit

A faster way to build and share data apps

Streamlit turns data scripts into shareable web apps in minutes.
All in pure Python. No front-end experience required.



<https://streamlit.io/gallery?category=llms>



[Explore](#)
[Pricing](#)
[Docs](#)
[Blog](#)
[Changelog](#)
[Sign in](#)
[Get started](#)

<h3>Generate images</h3> <p>Models that generate images from text prompts</p> <p> stability-ai/stable-diffusion stability-ai/sdxl stability-ai/stable-diffusion-inpainting </p>	<h3>Edit images</h3> <p>Tools for manipulating images.</p> <p> tencentarc/gfpgan sczhou/codeformer rossjillian/controlnet </p>	<h3>Restore images</h3> <p>Models that improve or restore images by deblurring, colorization, and removing noise</p> <p> tencentarc/gfpgan sczhou/codeformer jingyunliang/swinir </p>	<h3>Caption images</h3> <p>Models that generate text prompts and captions from images</p> <p> salesforce/blip andreasjansson/blip-2 yorickvp/llava-13b </p>
<h3>Get embeddings</h3> <p>Models that generate embeddings from inputs</p> <p> andreasjansson/clip-features daanelson/imagebind replicate/all-mpnet-base-v2 </p>	<h3>Upscale images</h3> <p>Upscaling models that create high-quality images from low-quality images</p> <p> nightmareai/real-esrgan jingyunliang/swinir mv-lab/swin2sr </p>	<h3>Use a language model</h3> <p>Models that can understand and generate text</p> <p> meta/llama-2-70b-chat meta/llama-2-13b-chat meta/llama-2-7b-chat </p>	<h3>Train a language model</h3> <p>Language models that you can fine-tune using Replicate's training API.</p> <p> meta/llama-2-70b-chat meta/llama-2-13b-chat meta/llama-2-7b-chat </p>
<h3>Chat with images</h3> <p>Ask language models about images</p> <p> yorickvp/llava-13b daanelson/minigpt-4 yorickvp/llava-v1.6-34b </p>	<h3>Use handy tools</h3> <p>Toolbelt-type models for videos and images.</p> <p> cjwbw/rembg lucataco/nsfw_image_detection fofr/audio-to-waveform </p>	<h3>Transcribe speech</h3> <p>Models that convert speech to text</p> <p> openai/whisper thomasmol/whisper-diarization vaibhavs10/incredibly-fast-whisper </p>	<h3>Generate music</h3> <p>Models to generate and modify music</p> <p> meta/musicgen riffusion/riffusion allenhung1025/looptest </p>
<h3>Generate videos</h3> <p>Models that create and edit videos</p>	<h3>Use a face to make images</h3> <p>Make realistic images of people instantly</p>	<h3>Generate speech</h3> <p>Convert text to speech</p>	<h3>Make 3D stuff</h3> <p>Models that generate 3D objects, scenes,</p>

Thank you 😊