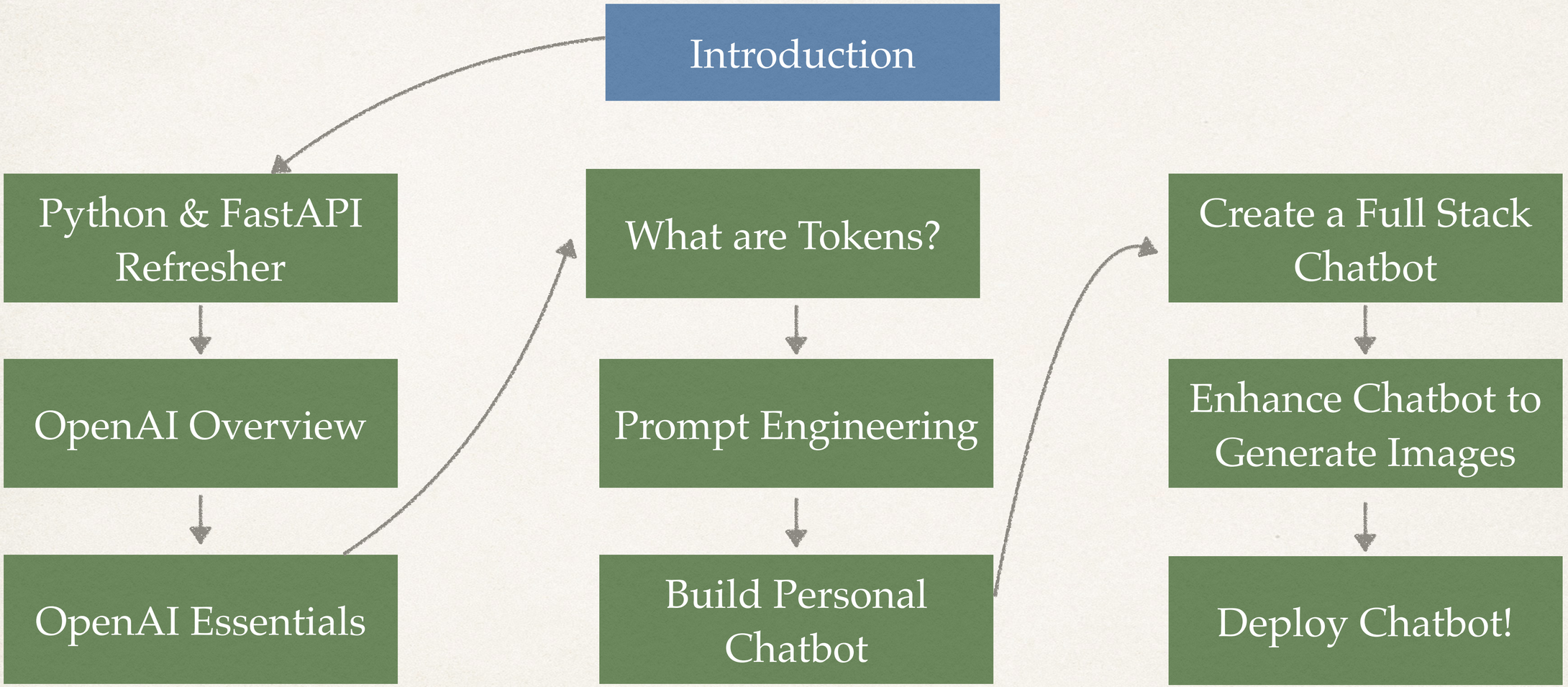
Course Content

Course Content!



How To Get The Most Out Of This Course

How To Get The Most Out Of This Course

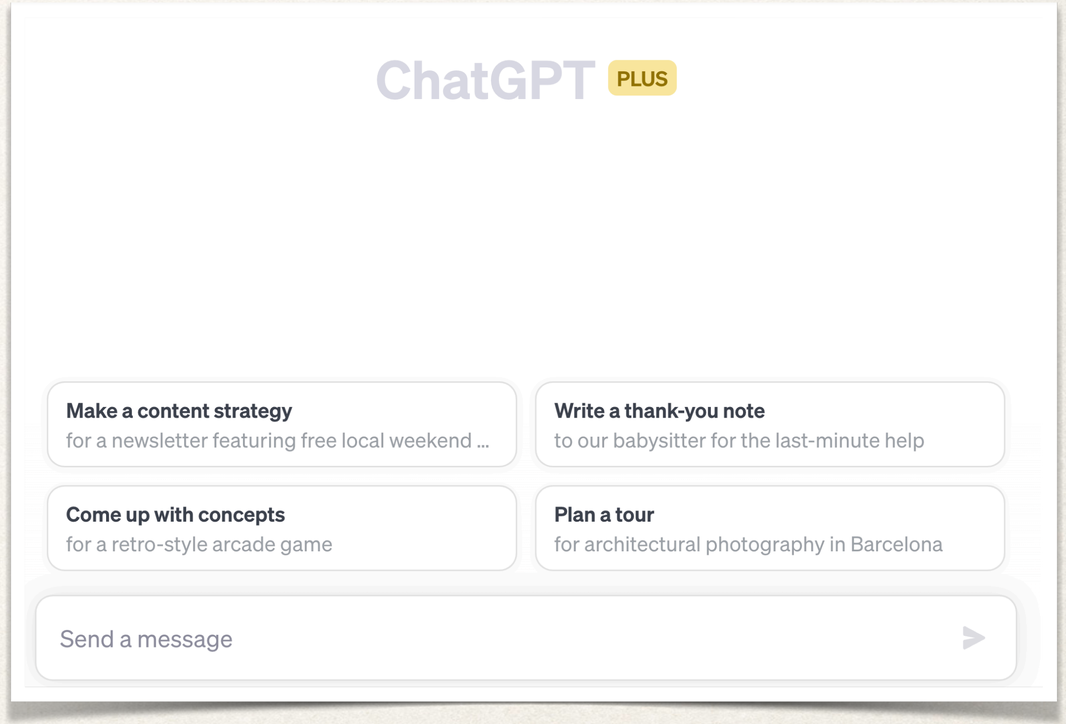
Watch all the videos	→	Control the speed of videos to be your pace
Take the exercises	→	Pause videos to take exercises before seeing solutions
Search & Practice	→	Advance on your own and search for additional information
Ask Questions	→	Ask in Q&A section!

What is OpenAI

ChatGPT vs OpenAI

ChatGPT

- ChatGPT is a product designed to generate human like responses to text input
- ChatGPT is a web application where you can type text and hopefully the application will generate a response
- Trained on massive text sets, including:
  - Articles
  - Social Media
  - Code and more..



How does ChatGPT work?

- To the user it is suppose to act like a virtual assistant
- You ask ChatGPT a prompt (text input) and get a response based on that
- Behind scenes ChatGPT is using a large language model:
  - GPT 3.5
  - GPT 4
- ChatGPT is a product from OpenAI

OpenAI

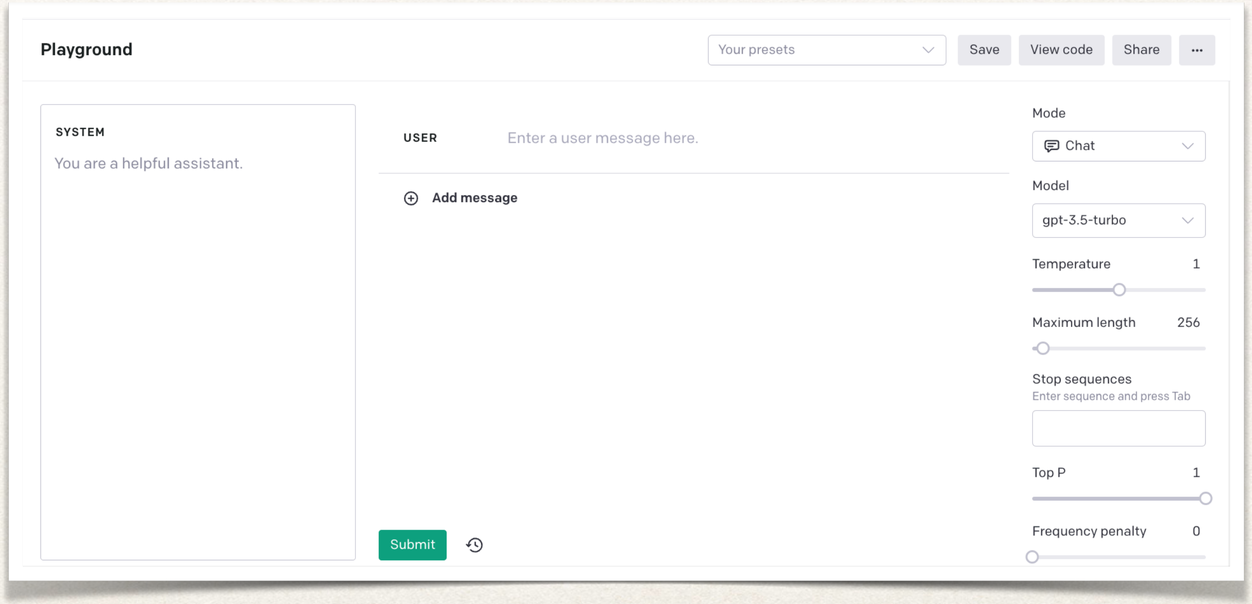
- At the core OpenAI is a AI (artificial intelligence) research and deployment company
- Mission is to ensure that AI benefits all of humanity
- Company that owns ChatGPT
  - ChatGPT launched in 2022 and is an AI tool originally built on top of a model known at GPT-3 (legacy). Provides a conversation interface that allows users to ask questions in natural language.

OpenAI API's

- OpenAI has public API's that can interact with their GPT models
- These are the same models that ChatGPT uses behind scenes
- The API's are ways for us to communicate with these models and be able to create our own Chatbots that are separate from ChatGPT.
- We can customize how our application interacts with these models to create a personalized Chatbot for our application

OpenAI Playground

- Is a web based platform that allows users to interact with different OpenAI models.
- This platform allows you to be able to change how these models interact based on arguments
- We will go over these arguments in a future video



OpenAI Cost

OpenAI cost

- OpenAI API's and ChatGPTs cost is different from one another (common mistake)
- ChatGPT currently is free for GPT-3.5 and costs ~$20 USD/month for GPT-4 models
- This is if you want to **only** use ChatGPT, not OpenAI API's.
- OpenAI API's are on a per use cost, however OpenAI has a free trial that will be enough for this course :-)

OpenAI cost

- OpenAI API's cost is based on the model and tokens

GPT-3.5 Turbo

GPT-3.5 Turbo is optimized for dialogue.

Learn about GPT-3.5 Turbo ↗

Model	Input	Output
4K context	$0.0015 / 1K tokens	$0.002 / 1K tokens
16K context	$0.003 / 1K tokens	$0.004 / 1K tokens

GPT-4

With broad general knowledge and domain expertise, GPT-4 can follow complex instructions in natural language and solve difficult problems with accuracy.

Learn about GPT-4

Model	Input	Output
8K context	$0.03 / 1K tokens	$0.06 / 1K tokens
32K context	$0.06 / 1K tokens	$0.12 / 1K tokens

OpenAI Tokens?

- We will dive more into what Tokens are in a later video
- A quick overview would be:
  - 1 token ~ 4 characters (3/4 a word)
  - 100 tokens ~ 75 words
  - 30 tokens ~ 1-2 sentences
  - 1 paragraph ~100 tokens

Model	Input	Output
4K context	$0.0015 / 1K tokens	$0.002 / 1K tokens
16K context	$0.003 / 1K tokens	$0.004 / 1K tokens

Cheap for practice use cases

OpenAI Tokens

OpenAI Tokens

- GPT works with tokens to determine input, outputs and price
- Models will split the user input and out into a bunch of tokens which typically is ~4 characters per token in English text

OpenAI Prompt Arguments

OpenAI Prompts

- Prompts are text inputs either from:
  - User (You)
  - Assistant (Response)
  - System (Behavior)
- Also known as messages
- Used to identify the log to add unique customization to User Inputs / prompts

OpenAI User Prompt

- User prompts are used as an input so the language model can answer
- You can type anything that does not add humanity fear

```
Summarize the book Computer Science by Eric Roby
```

```
If I wanted to learn Python, what is the best way to learn?
```

```
Please create a Python script that automatically empties the trash on my PC
```

```
What is 10*4+3-2*100
```

OpenAI System Prompt

- System prompts are used to steer the behavior of the AI model
- By default, the system prompt is “You are a helpful assistant”
- This allows the model to know to answer the user prompt in a generic helpful way

System Prompt

```
You are a Python tutor AI, completely dedicated to teach users how to learn
Python from scratch. Please provide clear instructions on Python concepts, best
practices and syntax. Help create a path of learning for users to be able to
create real life, production ready Python applications.
```

OpenAI User / System Prompt

- How does User & System prompts work together?

User Prompt

```
Write me a sentence to help me be in a good mood
```

Response

```
"Thy heart is but a vessel, wherein thy joy may flow abundantly and fill thy
soul with boundless delight."
```

System

```
You are Shakespeare
```

OpenAI Assistant Prompt

- Prompts that the AI model as created (responses)
- We can customize these to keep a history of actions

Assistant Prompt

```
Spurs won the 2005 NBA Championship
```

User Prompt

```
Who was on the team?
```

OpenAI Models

OpenAI Models

- AI program that analyzes datasets to find patterns and then be able to make predictions

OpenAI Models	
GPT-3.5	Understand and generate natural language
GPT-4	Understand and generate natural language (improvement over GPT-3.5)
DALL.E	Model that can generate and edit images based on natural language prompt
Whisperer	Model that can convert audio to text
Embeddings	Set of models that can convert text to numerical form
Moderation	Fine-tuned model that can detect whether text is unsafe

OpenAI Playground

OpenAI Playground

- Web-based platform that allows users to interact with OpenAI models
- Sandbox environment where you can see the response from different models based on the text provided
- Excellent tool to test models before implementation into your own API
- Helps with learning and experimenting to learn all the capabilities and limitations of OpenAI models

Chatbot Overview

Chatbot Overview

- We will be creating a full stack application using Python as the primary programming language
- OpenAI API Endpoints to create our Chatbot
- FastAPI to create our application's API endpoints
- Jinja2 for the User Interface to have a Full Stack application

Prerequisites

- Must create an OpenAI account
  - This is used to get our secret key to successfully call our models
- Be familiar with an IDE, this course will use PyCharm but you can accomplish the exact same application using VSCode or your preference
- Having some understanding on HTML will help when we create our front end / user interface for our application

Alright, lets dive in!

OpenAI Access Key

OpenAI Access Key

- An API key that is a unique code that identifies only your requests.
- Do not share this key with others
- This key represents you within OpenAI
- Do not deploy or commit this key to a remote repository

Environmental Variables

- User defined value that affects the process of an application
- We will setup our OpenAI access token as an environmental variable within our local application and when we deploy it
- That will keep our key safe and not hardcoded within our application

Create our Chatbot!

How to get started with our Chatbot

- First thing first:

```
pip install openai
```

- This will download all dependencies to be able to use OpenAI within our application
- We then need to setup our Environment Variable for OpenAI

```
export OPENAI_API_KEY=<KEY>
```

May have to do this more than once throughout the course

How to get started with our Chatbot

- We then start to write some code…

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }]
)

print(response)
```

Set The Model

Set System Prompt

Model Used

Assistant Content

Token Usage

How to get started with our Chatbot

- Add user prompt

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'user',
        'content': 'Who won the NBA championship in 2005?'
       }]
)

print(response)
```

Responds with the Spurs

Chatbot Messages

OpenAI Prompts

- Prompts are text inputs either from:
  - User (You)
  - Assistant (Response)
  - System (Behavior)
- Also known as messages
- Used to identify the log to add unique customization to User Inputs / prompts

Chatbot Messages

- Original Prompt - You cannot ask questions on the answer

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'user',
        'content': 'Who won the NBA championship in 2005?'
       }]
)

print(response)
```

Responds with the Spurs

- We need to set an assistant message (which would be the response)

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'assistant',
        'content': 'The Spurs won the 2005 NBA championship'
    }, {
        'role': 'user',
        'content': 'Who was on the team?'
       }]
)

print(response)
```

Responds with the players on the Spurs

Change behavior using System Prompt

- Generic System Prompt

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'user',
        'content': 'What was the greatest product of all time?'
       }]
)

print(response)
```

Responds with iPhone, Ford Motors, etc

Change behavior using System Prompt

- System Prompt that effects the behavior

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are the CEO of Apple'
    }, {
        'role': 'user',
        'content': 'What was the greatest product of all time?'
       }]
)

print(response)
```

Only responds with iPhone (due to being bias)

Configure Temperature

OpenAI Temperature

- Temperature is all about randomness
- In general words, the lower the temperature the more likely the model will choose words with a higher probability of occurrence
- If you are looking for a single solution (math problem) the best bet may be a lower temperature
- If you are wanting something creative (a story) a higher temperature may be a better choice
- Scale that starts at 0 and ends at 2

Change Temperature

- Change temperature to see different responses

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'user',
        'content': 'Write me a 3 paragraph bio'
       }],
    temperature=1.5
)

print(response)
```

Return just the content

- Pull the content out of the response

```
import openai

response = openai.ChatCompletion.create(
    model='gpt-3.5-turbo',
    messages=[{
        'role': 'system',
        'content': 'You are a helpful assistant'
    }, {
        'role': 'user',
        'content': 'Write me a 3 paragraph bio'
       }],
    temperature=1.5
)

print(response['choices'][0]['message']['content'])
```

Create Chat Log & History

Chat Log

- Implement our own Chat Log that will keep track of the entire history of our conversation with our bot.
- This will allow us to be able to ask questions on questions we previously asked
- One step closer to getting a full functioning chatbot implemented within our application

Chat Log Implementation

```
import openai

chat_log = []

while True:

    user_input = input()

    if user_input.lower() == 'stop':
        break

    chat_log.append({'role':'user', 'content': user_input})

    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=chat_log,
        temperature=0.6
    )

bot_response = response['choices'][0]['message']['content']
chat_log.append({'role':'assistant', 'content': bot_response})
print(bot_response)
```

Add our own FastAPI Endpoint

Implementing FastAPI into our Application

- First thing first:

```
pip install fastapi uvicorn python-multipart
```

- This will download all dependencies to be able to use FastAPI within our application
- First step in creating a web application chatbot

FastAPI Implementation

```
import openai
from fastapi import FastAPI, Form
from typing import Annotated

app = FastAPI()

chat_log = []

@app.post("/")
async def chat(user_input: Annotated[str, Form()]):

    chat_log.append({'role':'user', 'content': user_input})

    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=chat_log,
        temperature=0.6
    )

bot_response = response['choices'][0]['message']['content']
chat_log.append({'role':'assistant', 'content': bot_response})
return bot_response
```

Create Our First Page

Implementing Jinja2 into our application

- First thing first:

```
pip install aiofiles jinja2
```

- This will download all dependencies to be able to use Jinja2 within our application
- We will be using Jinja2 to create our User Interface for our users and clients

Jinja2 Implementation

layout.html

```
<!DOCTYPE html>

<html lang="en">

  <head>
    <meta charset="UTF-8">
    <title> Chatbot </title>
  </head>

  <body>
     Welcome!
  </body>

</html>
```

- Create a basic HTML file that we will be able to run on the browser
- Prints "Welcome" in the top left hand corner of the browser
- We will be implementing new features and CSS to this HTML file

FastAPI Implementation

```
import openai
from fastapi import FastAPI, Form
from typing import Annotated
from fastapi.templating import Jinja2Templates
from fastapi.responses import HTMLResponse

app = FastAPI()
templates = Jinja2Templates(directory="templates")

chat_log = []

@app.get("/", response_class=HTMLResponse)
async def chat_page(request: Request):
    return templates.TemplateResponse("layout.html", {"request": request})
```

The chat_page function will start our layout.html file

Chat Responses

FastAPI Implementation

```
import openai
from fastapi import FastAPI, Form
from typing import Annotated

app = FastAPI()

chat_log = []

@app.post("/", response_class=HTMLResponse)
async def chat(request: Request, user_input: Annotated[str, Form()]):

    chat_log.append({'role':'user', 'content': user_input})
    chat_response.append(user_input)
    response = openai.ChatCompletion.create(
        model='gpt-3.5-turbo',
        messages=chat_log,
        temperature=0.6
    )

bot_response = response['choices'][0]['message']['content']
chat_log.append({'role':'assistant', 'content': bot_response})
chat_response.append(bot_response)
return templates.TemplateResponse("layout.html", {"request": request})
```

Jinja2 Implementation

home.html

```
<div class="card-body">
  {% for response in chat_responses %}
  <div>
  - {{response}}
  </div>
  {% endfor %}
</div>
```

- Start the for each loop
- Print the response
- Complete for each loop

DALL-E Overview

DALL-E Overview

- Created by OpenAI so we do not have to do anything extra to implement within our application.
- AI System that can create realist images and art from a description using natural language.
- Can create original, realistic images and art from text descriptions. It combines concepts, attributes and styles.

DALL-E Overview

User Prompt

```
Create a cartoon duck wearing glasses
```

```
response = openai.Image.create(
    prompt=user_input,
    n=1,
    size="1024x1024"
)
```

Deployment Overview

Deployment Overview

- We will be deploying on AWS (Amazon Web Services)
- We will be using HTTPS to pass encrypted data over the internet
- Have our own URL so anyone can use our application

How do we plan to do this?

AWS Lambda

- Are known as serverless
- They are known as serverless because you do not have to setup servers (AWS does the server setup)
- You write code and upload it on a Lambda and after a few configurations, it will be up and running
- Automatic Scaling
- Pay only for the amount being used (free with the Free Tier)

Why use AWS Lambda

- You do not have to recreate the wheel.
- If in the future we want more compute space or a backup, we are already in AWS and can be configured easily.
- Do not have to use a remote repository, we can deploy a .zip file
- Configuring environment variables are easy.
- Safe and secure being backed up by Amazon.

Thank You!

Let’s Stay In Touch

- Feel free to contact me :-)
  - codingwithroby@gmail.com
  - https://www.youtube.com/@codingwithroby