**Udemy: Introduction to OpenAI API & ChatGPT API for Developers**

**Section-1: Introduction**

**Chapter-1:**

Machine Learning is not the same as AI.

It is a subset of AI

Refers to the development of algorithms that mimic the human brain.

Machine Learning is a technique to achieve Artificial Intelligence.

Machine Learning Model is a result of training a machine learning algorithm on a particular dataset.

Prompt is an input we give to a Model. Prompt -> Model -> Output

Token is chunk of data.

**Chapter-2:**

Created postman account using google account.

**Chapter-3:**

Created platform.openai.com using google account.

**Chapter-4:**

Created API KEY in platform.openai.com:

**Chpater-5:**

Created GET method in postman

A screenshot of a computer

Description automatically generated

**Chapter-6:**

Created POST method in postman

A screenshot of a computer

Description automatically generated

**Chapter-7:**

Setting up billing information

**Chapter-8:**

Overview of HTTP

**Chapter-9:**

JSON format explained: Booleans and numbers don’t have doble quotes.

**Chapter-10:**

Understanding GPT:

Generative Pre-trained Transformer

GPT is an AI Model that is pretrained and can understand text and generate new text.

**Chapter-11:**

Difference between OPEN AI and ChatGPT

OpenAI is a company and ChatGPT is one of their products including DALL-E, WHISPER.

**Section-2: Overview of the OpenAI models and Capabilities**

**Chapter-12: selection overview**

Chat GPT uses gpt-3.5-turbo model.

Dall-E: A model that can generate and edit images given a natural language prompt

Whisper: That can convert audio into text.

GPT-3 is older version and has obsolete prompts.

**Chapter-13: What are completions**

Completion: Generating a new text that is like the text which was used to train the AI.

E.g: Prompt: Write a title for a socio fantasy story

Completion: The warrior hero from the Jambu Dweepa

--in a way, completion is an output given by Open AI to the prompt we give to it.

**Chapter-14: Open AI Playground**

Introduction about System role, assistant role, and user role… how system instruction changes the behavior of completion for given prompts.

**Chapter-15: Chat completions with GPT 3.5(gpt-turbo3.5)-ChatGPTAI**

Created POST method with system and user messages.

**Chapter-16: Chat completions with GPT 4**

Created another 'Chat Completion GPT4' post method and got successful response.

**Chapter-17: gpt3.5-turbo vs GPT 4**

GPT4 has more tokens processing capabilities.

**Chapter-18: Text completion with gpt3.5(text-davinci-003)**

Created Chat\_Completion\_gpt3.5Turbo-davinci-model, but used turbo3.5 model.

**Chapter-19: gpt3.5-turbo vs text-davinci-003**

Text-davinci-003 is traditional completions… also it is better to try different models to check which one is giving best completions for us.

**Chapter-20: Image generations with DALL-E**

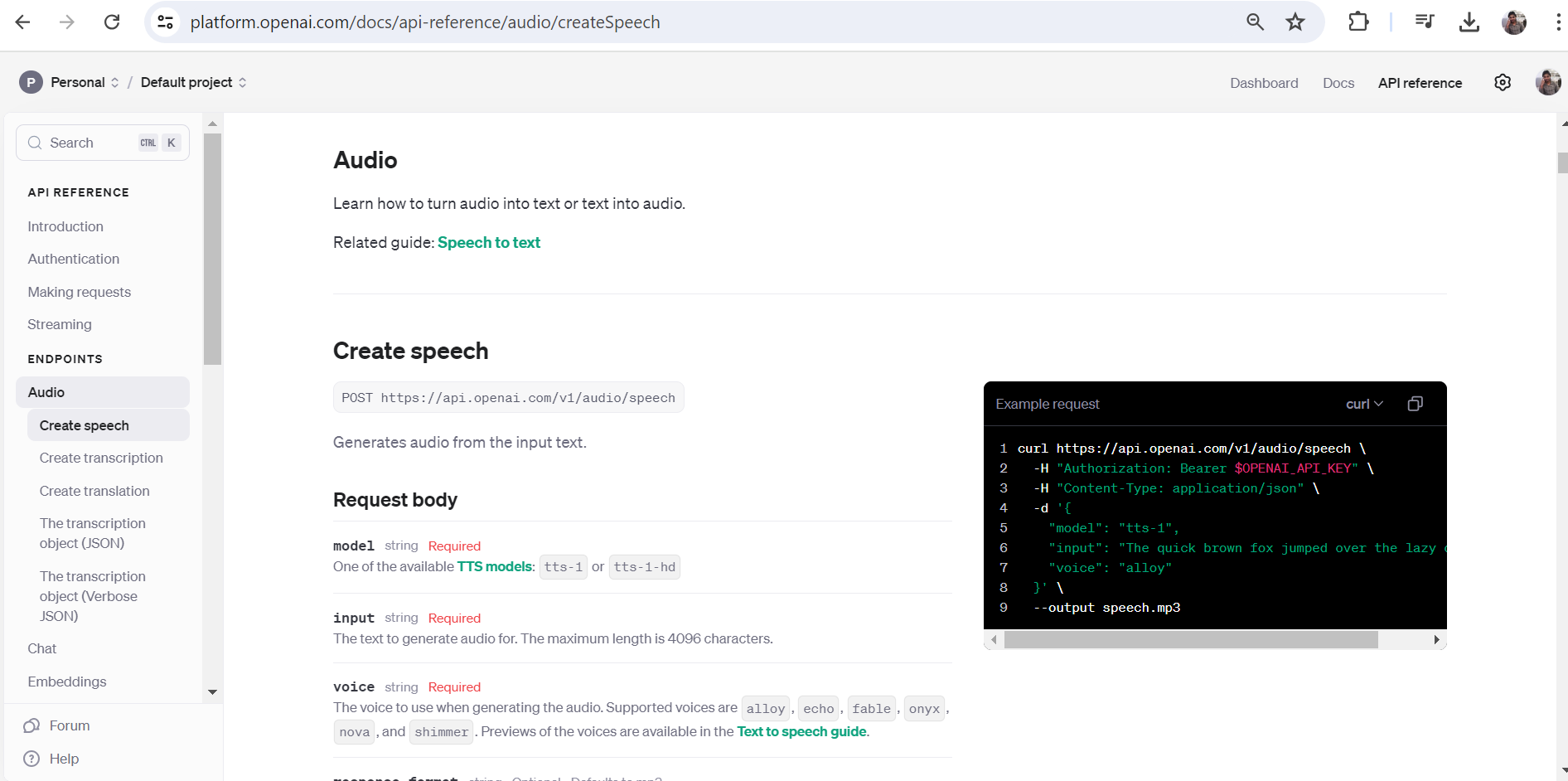
In file "Image\_Completion\_DallE3" using DALL-E in post method created an "avocodo eating a knife"

A screenshot of a computer

Description automatically generated

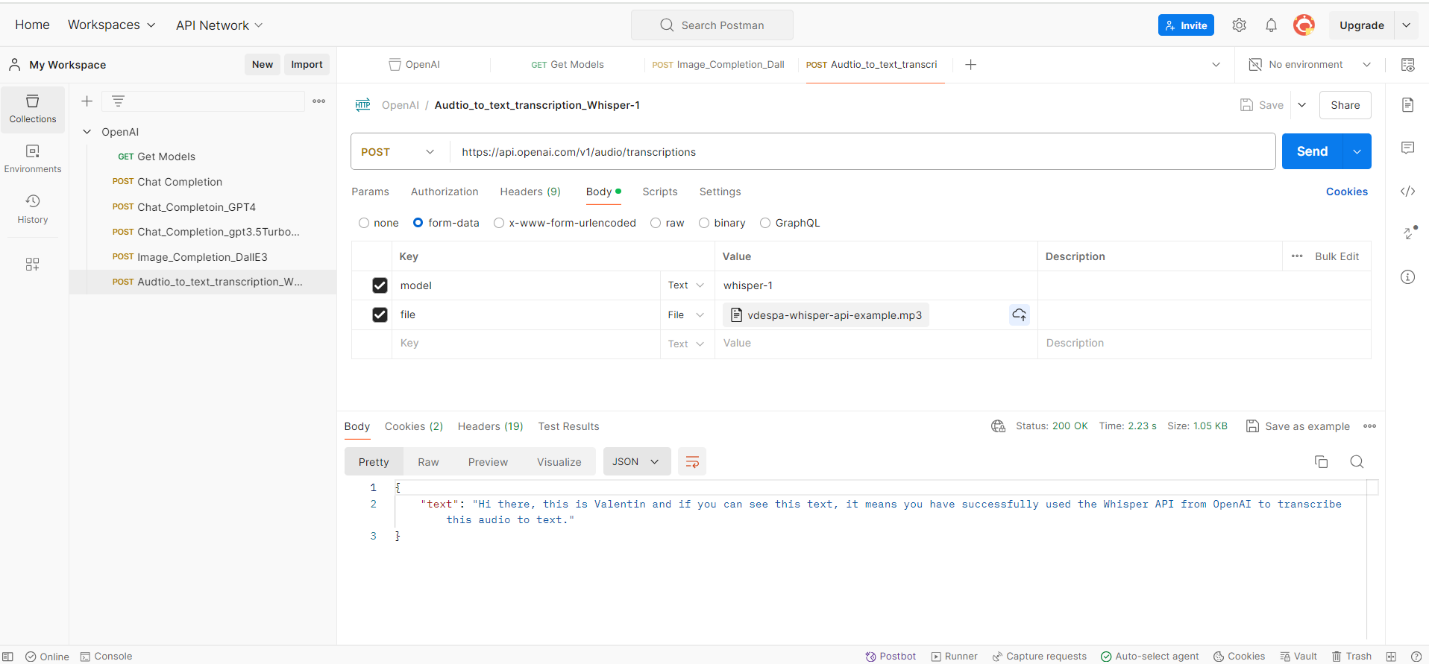
**Chapter-21: Speech recognition with Whisper API(audio to text)**

We learnt from documentation (platform.openai.com/docs, in Audio end point, create transcription) this has only one model **whisper-1**, its not available in playground so we directly call the endpoint from postman.



Checked if the model is actually available for us in postman by searching 'whisper-1' in 'GET Models' output.

Created a new POST request 'Audtio\_to\_text\_transcription\_Whisper-1' like below, uploaded the file from local machine and got the response as shown in screen.



**Chapter-22: Prompt Design**

The shorter the prompt, the vague the results. Better prompt gives, better results. Just did some practice in OpenAI play area on chat prompts.

**Chapter-23: Adjusting Temperature**

Increasing the temperature may give more wider variety of values but making it 2 may give gibberish values at times.

**Chapter-24: Tokens and Problems**

Model: gpt-3.5-turbo, Usage: $0.002/1k Tokens. 1K tokens is ~700-750 words excluding spaces and special characters.

**Chapter-25: OpenAI API status**

Open AI may be unavailable at times, we can check the status here <https://status.openai.com/>

**Section-3: GPT 3.5 models - Chat GPT API**

**Chapter-26: Section Overview:**

In this section, we will explore in more depth the gpt-3.5-turbo model, which powers ChatGPT, and which is the most advanced model from OpenAI.

**Chapter-27: gpt 3.5-turbo model basics**

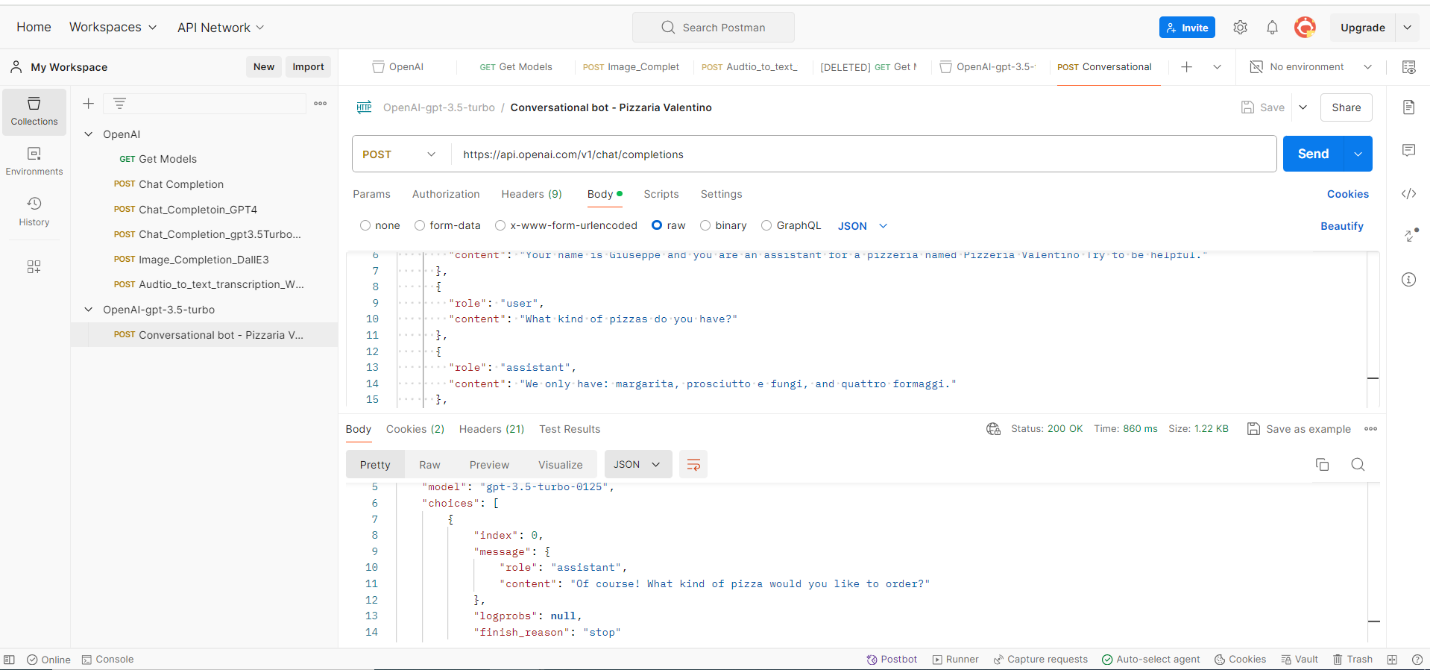
Again, touched the points like Assistant, System, and User roles

**Chapter-28: Instructing the model with examples.**

We duplicated the OpenAI collection, except Chat completion, removed all other programs.

Added the API key in order to make the new collection work.

Modified the chat completion by giving instructions to System, User, and Assistant prompts.



**Chapter-29: Create a summary of Text (multi-line prompt).**

We cant give multiline text inside user content in the BODY section. To overcome the issue, we user the pre-script or script section to give the multiline prompt with basic syntax, then we will be able to get the summary of multiline text/prompt.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Observe the script is placed inside "Pre-Request" area, not in "Post-Response" area.

**Chapter-30: Streaming output (chat typing effect)**

Instead of waiting for output to appear after few seconds, if we want to see the sub mill-second responses for each token, we may include property "stream" : true to get the data. By default, this "stream" option is set to false, so that we get final completion from Open AI in one go instead of tokens.

We can make this change to our existing "Multi-line Summary" program.

A screenshot of a computer

Description automatically generated

**Chapter-31: Limiting the length of completion**

max\_tokens is the option to limit the completion length. It’s a hard limit, model will give only mentioned max\_tokens.

A screenshot of a computer

Description automatically generated

**Chapter-32: Stop Sequences**

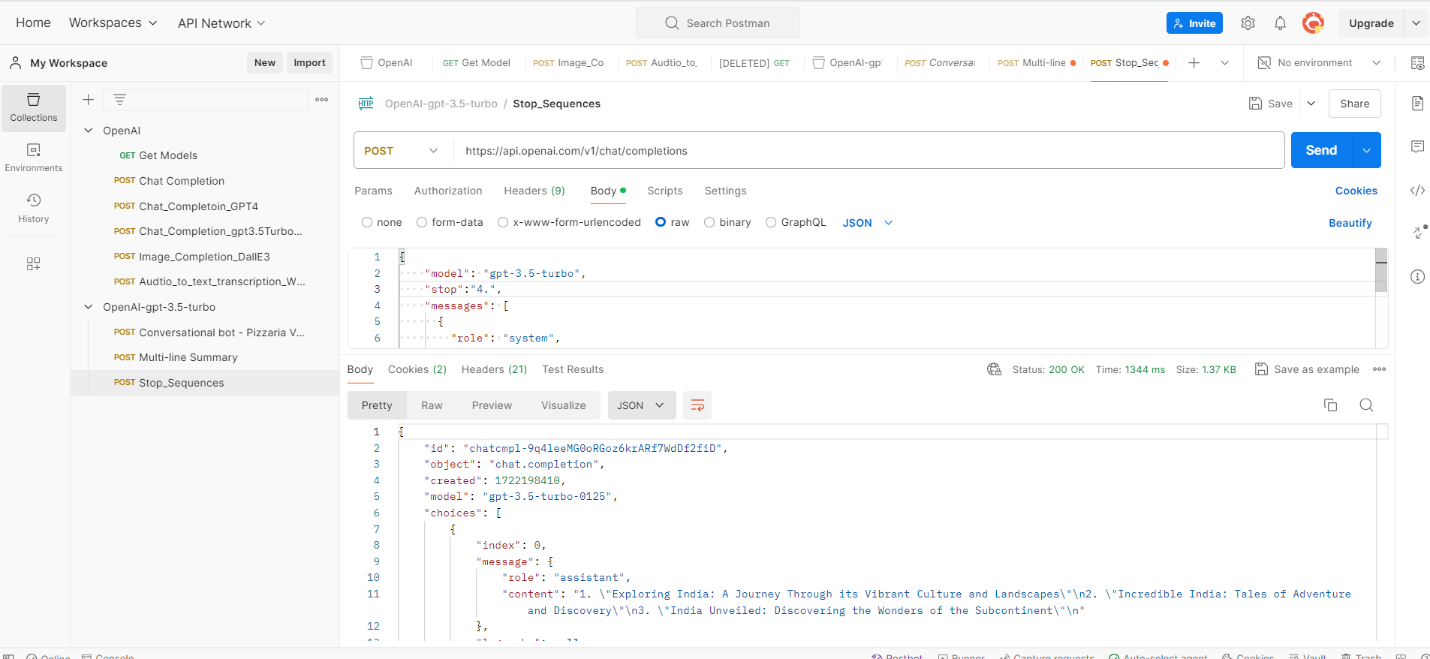
Whenever we want the completion to stop at any point, we can use "stop":"<<keyword>>". The moment the response hit the <<keyword>> it stops irrespective of what we give in our prompt.

E.g: Without stop.

A screenshot of a computer

Description automatically generated

With "stop": "4." --> as soon as it hits the keyword "4.", it stops the results.



Though we have asked for top-10 travel titles, it stopped as soon as it hit "4." . This way, we can limit our output by anywhere we want to stop specifically.

**Chapter-4: Build your own ChatGPT clone**

**Chapter-33: Section Overview**

We generate ChatGPT clone using open source APIs. We develop a web application using web development framework called Next.js.

**Chapter-34: Installing Required Tools**

NodeJS will allow our application to run locally. --> Installed NodeJS from web. (Use the LTS version, not current version). <https://code.visualstudio.com/docs/?dv=win64user> <https://nodejs.org/en>

Open a powershell window and run below commands.

A screenshot of a computer program

Description automatically generated

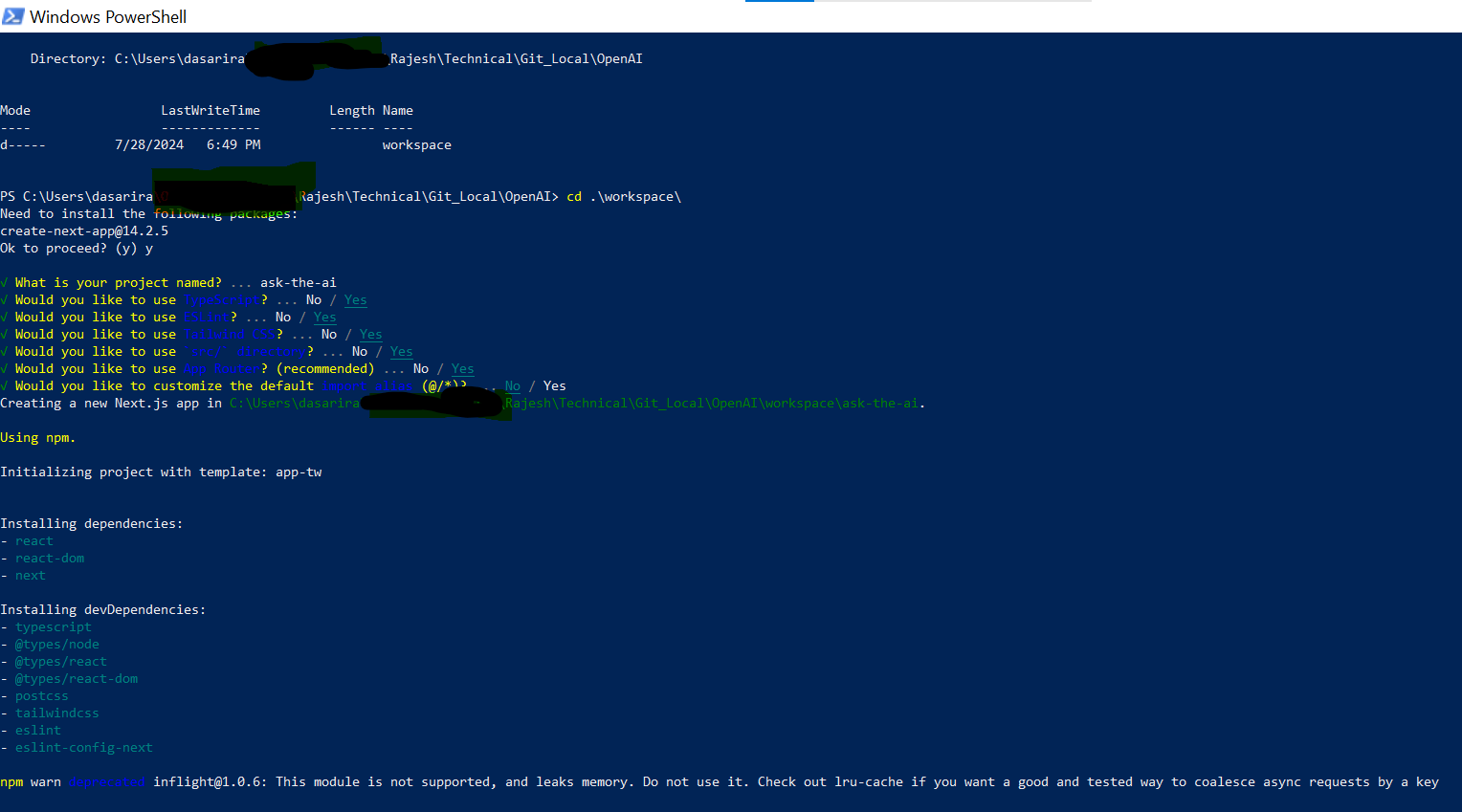
**Chapter-35: Creating a new Next.js project**

Create a directory in local drive to save all these projects in one location.

C:\Users\dasarira\OneDrive - HP Inc\Rajesh\Technical\Git\_Local\OpenAI\workspace

Cisit the nextjs.org page <https://nextjs.org/> and copy the command npx create-next-app@latest

This will ask us to give the project a name, then give few NO/YES questions.



We no longer need the powershell. We can open the powershell terminal from Visual Studio and run command as shown below. It will open a page in <http://localhost:3000>

A screenshot of a computer program

Description automatically generated

We may see the following webpage on localhost:3000 port after running "npm run dev" command. Now our program is successfully running. If we want we can simply Ctrl+C our application and stop it from powershell terminal from where we ran the program.

A screenshot of a computer

Description automatically generated

**Chapter-36: Creating the user interface using ChatGPT**

As per our requirement, we need to start making changes to src/app/page.txt. By default we have lot of code, but we don’t need all that code in the page. We can ask chatGPT to provide us the necessary code as shown in screens below and update out page.txt accordingly.

Use the "prompts.txt" to get the code from chat.openai.come… final /src/app/page.tsx file should have the code from page.tsx file attached. If the webpage is not working or page is throwing any error when you save it, restart the application.

Now the page should look like below.

A screenshot of a computer

Description automatically generated

**Chapter-37: Creating an API endpoint**

First create an API inside source/app…with name api/openai/route.ts

**Chapter-38: Integrating OpenAI**

Integrated openAI libraries by running "npm install openai".

**Chapter-39: Making UI Improvements**

Since the AskAI button is keep on waiting, changed it to processing once we submit the request and separated text boxes for each request and response.