Next JS – Open AI – Chatbot – With RAG.

Tech Stack:

* Next JS
* Type Script
* Tailwind CSS
* OpenAI Chat Completion API

Tutorial:

1. <https://www.youtube.com/watch?v=gbZTEfo7BBo> (Done)
2. <https://www.youtube.com/watch?v=bx0oryJCpSk> (Done)
3. <https://www.youtube.com/watch?v=y5eLukU5ur8> (17 mins.)

Sequence:

1. Create Project Directory (Practiced using Git Bash)
2. Create Next JS Project (npx create-next-app@latest) with normal settings
3. Create React – Icons (npm i react-icons)
4. Created ChatBot Component
5. Created a use-effect hook with use client. (Learn more)
   1. Importing useState
   2. This is done for conditional rendering
6. Created a component for ChatBot Message.
7. Installed Open AI (npm install openai)
8. Created OpenAI Platform Account
9. Created a Secret KEY (API)
10. To Chat to OPEN AI, we have to create separate folder for Server Actions in Next.JS
11. Adding RAG to the System

Notes:

* Main page is page.tsx and to simplify the process we can delete everything in between the main tags
* Create components directory to store all components and then create the folder for separate components like ChatBot
  + Note the difference between rafc, rafce and creating components like export default function Name() { return ….. }
  + Note that the components we created needs to be called on the page.tsx to be displayed on the app.
  + Note that by default in Next JS react hooks are server components so we have to tell Next JS that it is a client component mentioning **use client**.
  + We used conditioning rendering based on the value of useState. If its sets to true it returns the divs or nothing! An arrow function alters the value of the useState so that it can create the toggle effect.
* Layout.tsx contains the meta data like title and others.
* Open AI documentation to setup is here: <https://platform.openai.com/docs/guides/chat-completions>
* See, how we created different things using the documentation like useState.
* Note the destructing done to display dynamic messages instead of hard quoted.
* Topics to be learned later (Type Script):
  + Props
  + Setting States
  + Array Maping
  + Arrow Functions
  + Destructuring
  + Spread operator …
* After setting all the functionality we talked to OpenAI
  + Saving the messages so that it can know what is the context.
  + Server actions in Next.JS
* Working with Open AI chat completion API is not that difficult. Didn’t take much to address. But it is important to understand what we have done with the documentation for better understanding.
  + For using API, payment is must! (Done and its working now!)
* RAG … What is?
  + Retrieval Augmented Generation combines retrieval based and generation based models.
  + Kind of hybrid model. Purpose is to improve the accuracy of the output by taken help from outsourced or more relevant data. (Extremely important concept!)
  + External DATA Sources: Database, JSON Files.
  + It has two phases
    - Retrieval phase: where based on search algorithm is used to find relevant content from the large data
    - Generation phase: where generative models are used to generate response based on user’s query and retrieved content
  + Benefits of RAG:
    - Improved accuracy
    - Relevance
    - Can be used in Service ChatBOTs etc.
  + Applications
    - Customer Support
    - Content Generation
    - Educational
  + Personal Notes:
    - Imagine Some System where chatbot can communicate with the database and get exactly relevant information.
    - This can be integrated anywhere. Even with Hanco, for example for complaint tracking, warrenty, etc.
    - Even in tutoring application or any software, more user specific text generation.
    - Fully integrated and customizable to E – Commerce Solutions as well.
      * E-commerce support chatbot
        + Have to think for Common platforms like wordpress shopify etc.
  + Popular RAG Models
    - Open AI GPT
    - Facebook AI’s RAG Model
    - Microsoft Turing NLG with Retrieval