

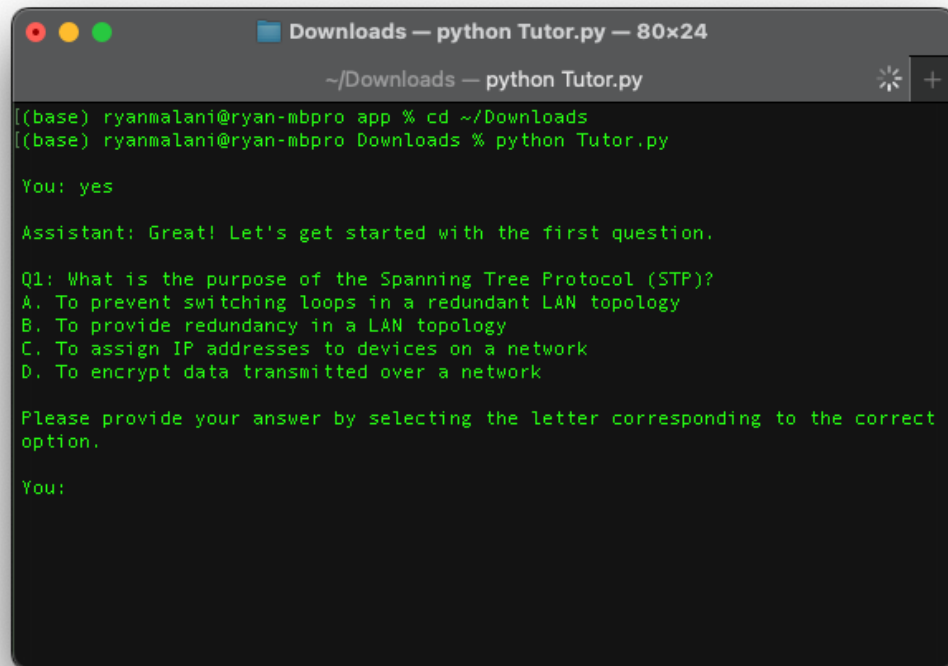
Building an AI-Powered Study Chatbot

by Ryan Malani, Shaun Laranjeira

📅 August 11, 2023

🕒 4 min read

As a result of some of our programs, our employees are required to obtain and maintain various certifications - most significantly the Security+ from CompTIA and CCNA [Cisco Certified Network Associate]. This has led us to scratch our heads at times due to the limited free study resources available for these popular certifications, or the difficulty searching and finding recommendations from others who have taken and passed the test. This sparked an idea internally to build a chatbot using OpenAI's gpt-3.5-turbo conversational AI large language model (<https://olivia-e-brown.medium.com/beginners-guide-to-openai-s-gpt-3-5-turbo-model-45bdce194d19>). Being that this model is comprised of an extremely large and diverse internet dataset, we decided to do some testing with a simple python script to see if it would make a good quizzing partner.

A terminal window titled "Downloads — python Tutor.py — 80x24" with a dark background and green text. The window shows the execution of a Python script that runs a quiz. The user has navigated to the Downloads directory and executed the script. The script prompts the user to answer a question about the Spanning Tree Protocol (STP). The user has responded with "yes". The script then displays the question and four multiple-choice options. The user is prompted to provide their answer by selecting a letter.

```
[(base) ryanmalani@ryan-mbpro app % cd ~/Downloads  
[(base) ryanmalani@ryan-mbpro Downloads % python Tutor.py  
  
You: yes  
  
Assistant: Great! Let's get started with the first question.  
  
Q1: What is the purpose of the Spanning Tree Protocol (STP)?  
A. To prevent switching loops in a redundant LAN topology  
B. To provide redundancy in a LAN topology  
C. To assign IP addresses to devices on a network  
D. To encrypt data transmitted over a network  
  
Please provide your answer by selecting the letter corresponding to the correct  
option.  
  
You:
```

Once we had validated it's capabilities for creating quiz questions and providing feedback on the correct answer, we were ready to explore how we could take our user unfriendly terminal script and turn it into a clean, modern, and performant human-centered-design.

Getting Started

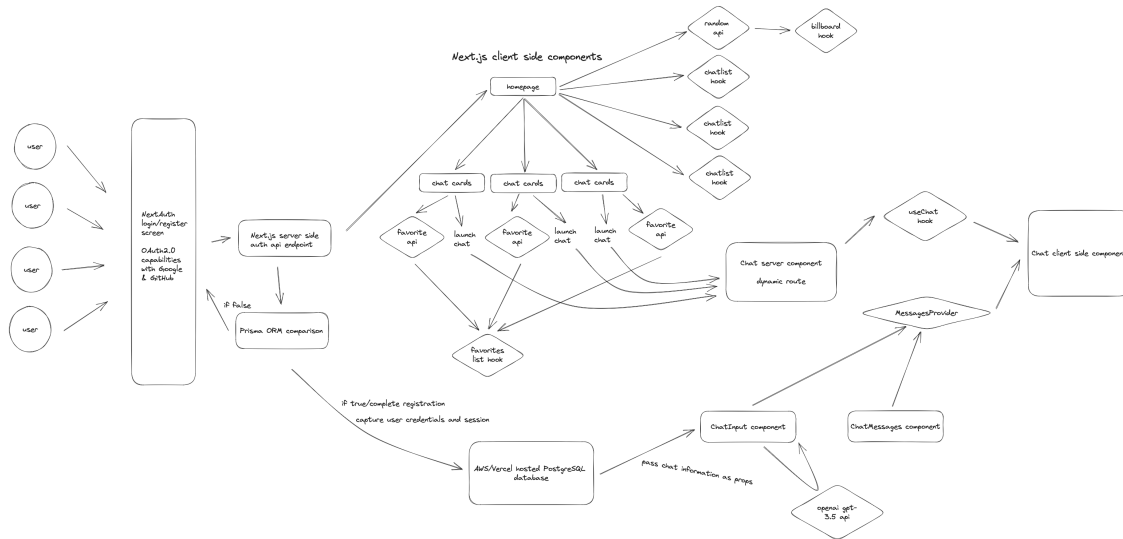
After conducting some initial research, we landed on the best-of-breed frontend tools and technologies to make this project a reality:

1. Next.js (<https://nextjs.org>) - This is a robust React Framework that is designed for creating full-stack applications [React Server and Client side components] while also building extremely quick thanks to Rust-based JavaScript tooling. Plus, we could use TypeScript and many of the other great features.
2. TailwindCSS (<https://tailwindcss.com>) - This is a CSS Framework that makes scaling with our team that will continue to work on the frontend of this app much easier by keeping the classes in the HTML markup.
3. shadcn/ui (<https://ui.shadcn.com>) - This is an open-source customizable component library for React/Next.js. The designs are clean, modern, minimalist, and reliable.
4. SWR (<https://swr.vercel.app>) - Built by Vercel to complement Next.js, we are using SWR - a tool for data fetching from React Hooks.
5. headlessUI (<https://headlessui.com>) - Built by TailwindLabs, this is an unstyled React/Vue UI component library that sits under many of our UI elements.
6. Docker (<https://www.docker.com>) - This is an obvious one; Docker allows us to develop locally and ensure that our code ships and works anywhere. Makes our build and deployment process much easier!

With all of that said, and our development process setup, we spun up a new local postgres database, dockerized Next.js app and started installing some packages with npm.

Defining the Architecture

To create the initial pilot, we decided to use the built-in functionality of Next.js client and server components. We outlined the architecture below as follows [using excalidraw (<https://excalidraw.com>)]:



Building the App

Being that Next.js is built off of [React.js](https://react.dev) (<https://react.dev>), we're using many of the familiar advantages in [React web hooks](https://react.dev/reference/react) (<https://react.dev/reference/react>) and [providers](https://react.dev/reference/react/createContext#returns) (<https://react.dev/reference/react/createContext#returns>).

There are then some additional great benefits of Next.js specifically, such as Dynamic Routing (<https://nextjs.org/docs/pages/building-your-application/routing/dynamic-routes>), which enables us to have a reusable chat interface that is passed props and data from our postgresQL database. NextAuth (<https://next-auth.js.org>) makes authentication services incredibly easy.

We use agile software development practices internally and DoD DevSecOps approved tools for the SDLC to include GitLab & Jira. This enables us to develop software at a fast pace, ensure we're delivering a quality product for the user, and implement modern development and shipping processes [CI/CD, merge requests, code reviews, etc.]

Using the Chatbot

It's incredibly simple- once authenticated, the homepage is rendered using webhooks for different categories of certifications and chat cards for the individual chats that when hovering over, can either be favorited and added to your list of favorites, or launched. When launched, the chatbot already has all of the necessary context and is ready to quiz the user. Once you say yes to the bot, it will begin the quiz. You can respond with simply a single letter for your answer choice, and it will give you a response of whether it was correct or not, the reason for the correct answer, and the next question to answer. Try it out for yourself [here](https://study.inflowfed.com) (<https://study.inflowfed.com>)!

Future Iterations

There are a variety of ways for us to expand on this project, and we're using a Jira Kanban board to track our items to work on. As of right now, we're looking to increase the reliability of the authentication services, implement more unit tests, improve user experience through the interface (adding skeletons to loading images, animations, etc) and add more support for certifications to the platform. The larger portions of the site that are being worked on currently are improving the infrastructure and architecture (transitioning some of our containerized services to AWS ECS and some of the backend services to FastAPI for improved latency) as well as improving our data collection on usage to improve the user experience through machine learning based recommendations (potentially using the now [open-sourced Twitter recommendation algorithm](https://github.com/twitter/the-algorithm) (<https://github.com/twitter/the-algorithm>)) upon login as opposed to the current randomized method. As we work on these areas of expansion, we'll continue to update here on our labs site. if you have any feedback or want to learn more about our capabilities in this area, feel free to [contact us](#).