**NANYANG TECHNOLOGICAL UNIVERSITY**

**School of Computer Science and Engineering (SCSE)**

**SC4052: Cloud Computing**

**Text

Description automatically generated**

**Academic Year: 2023-2024, Semester 2**

**Assignment 2**

|  |
| --- |
| **Teoh Xi Sheng (U2120456L)** |

Contents

[1. Build a Quiz Generator with Google Gen-AI 3](#_Toc162714336)

[2. Using Google Gen-AI to Generate Cloud Computing Quizzes 3](#_Toc162714337)

[3. Using Nemobot with OpenAI API to Generate Quizzes 5](#_Toc162714338)

[4. Exploring Browser-based LLM (WebGPU) 7](#_Toc162714339)

[Appendix A: Nemobot Code Reference 8](#_Toc162714340)

# 1. Build a Quiz Generator with Google Gen-AI

The following is the summary of the tasks for creating this quiz generator:

**Backend**

This backend server will handle user requests, process requests by forwarding them to Google’s Vertex AI, and return the responses in JSON format.

1. Using Google’s Cloud Shell editor, build a RESTful web server using the Flask module from Python to handle POST and GET requests.
2. Program the web server such that it will handle parameters from the URL.
3. Process the parameters such as “topic” and “num\_q” together with the prompt using the Vertex AI module from Python.
4. The module will send a request to the Vertex AI from Google Cloud and return the quizzes in JSON format.
5. Host the web server on Google Cloud so requests can be sent over the internet.
6. Backend URL: <https://quiz-generator-2v42b5qzta-de.a.run.app/>

**Frontend**

This frontend server allows creators to host quizzes generated by Vertex AI and learners to participate in live quizzes hosted like “Kahoot”.

1. Upon logging into [Quizaic](https://quizaic.com/), the creator can generate a quiz template by inputting the various parameters such as “Topic”, “Number of Questions”, and a “Custom Generator URL”, which will be the backend [URL](https://quiz-generator-2v42b5qzta-de.a.run.app/) that we have created previously.
2. After submitting the template, the creator can expect a quiz template created by Vertex AI in a user-friendly format.
3. The creator can then host a live quiz using the template for participants to participate in.

While building this application seemed relatively straightforward, we can appreciate the multiple cloud services provided by Google that made this possible.

|  |  |
| --- | --- |
| **Google Cloud Service** | **Usage** |
| Cloud Shell | Provides users with a cloud development environment with CLI to interact directly with the services. |
| Cloud Build | Executes and builds the application into artifacts such as Docker containers |
| Artifact Registry | After building the app, the container will be hosted in the registry for deployment. |
| Cloud Run | We use this service to deploy the instance of the container application to the internet. |
| Cloud Storage | This is mainly used to store application data. For example, we can store the Vertex AI prompt that we have prompted for edits later. |
| Vertex AI | This is a SaaS like OpenAI’s ChatGPT. We interact with it using API call. |

# 2. Using Google Gen-AI to Generate Cloud Computing Quizzes

There are two possible ways to generate cloud computing quizzes using Google’s Vertex AI.

**Method 1: Quizaic with Vertex AI.**

We can host a fully functional quiz using Quizaic as front-end and the backend server connected to Vertex AI. This method will be using the Vertex AI parameters that have been pre-configured, which means the creator has no control over settings such as the language model and temperature.

**Method 2: Using Vertex AI directly.**

If we would like to generate the quiz based on a certain specific knowledge, we can make use of the Vertex AI but feeding in some contexts. For this experiment, we will feed in texts from slide 4-12 from lecture 5 – Cloud Security.

A screenshot of a computer

Description automatically generated

As we use the same prompt but setting the temperature to 0.2, it reduces the randomness of the response. It produces a more deterministic and conservative outputs despite feeding in a small context. In the scenario below, we can observe that both questions are asking similar questions, unlike when temperature = 1.

A screenshot of a computer

Description automatically generated

# 3. Using Nemobot with OpenAI API to Generate Quizzes

Using OpenAI API and Nemobot, we can create and perform prompt engineering by customising our LLM functions for specific application. Using Nemobot, I have created the following LLM functions so that the chatbot can generate quiz based on a given knowledge: Detailed code can be found in Appendix.

// Detect the content given, learn the knowledge, and add in other knowledge on a similar topic so that it can be used to generate quiz questions.

const knowledgebase = parseKnowledgeBase({messagge});

// Detect the type of questions to generate

const questionType = parseQuestionType({message});

// Detect the number of questions to generate

const questionCount = parseQuestionCount({message});

// Generate the quiz questions

const generateQuestions({knowledgeBase, questionType, questionCount});

The initial prompt wasn’t accurate despite letting it know what it should detect and response.

For example, in parseQuestionType({message}):

A screenshot of a computer

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Nemobot: What type of questions do you like to generate?** | | |
| **User Response** | **Nemobot Interpretation** | **Expected Interpretation** |
| “t f” | “true/false” | “true/false” |
| “multiple-choice type” | “multiple-choice” | “multiple-choice” |
| “a b c” | “Open-ended” | “multiple-choice” |

In parseQuestionCount({message}):

A screenshot of a quiz

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Nemobot: What type of questions do you like to generate?** | | |
| **User Response** | **Nemobot Interpretation** | **Expected Interpretation** |
| “eleven” | “eleven” | “eleven” |
| “ten + one” | “one” | “eleven” |

With the help of unit testing, I have improved the accuracy and refined the functions by giving examples and requesting that reasoning to be included in its response (as suggested by the Nemobot tutorial).

Improved parseQuestionType({message}):

A screenshot of a computer

Description automatically generated

Improved parseQuestionCount({message}):

**A screenshot of a quiz

Description automatically generated**

Here is how the Nemobot perform. Full demo can be found in the screen recording in the zip folder.

A screenshot of a chat

Description automatically generated A screenshot of a chat

Description automatically generated

# 4. Exploring Browser-based LLM (WebGPU)

Instead of relying on cloud-based storage and execution of LLMs, advancements in WebGPU technology now enable clients to access and run LLMs directly on their local devices without internet connectivity. WebGPU allows web browsers to utilize the device's GPU as a hardware accelerator, enabling the models to be executed without the need for cloud GPU processing.

The [WebGPT](https://github.com/0hq/WebGPT) project doesn’t work on my local machine or the web, and the model was responding with “#”, similar to the issue raised [here](https://github.com/0hq/WebGPT/issues/4). The issue persists despite using the GPT2 1.5B model.

A black and white text

Description automatically generated

However, as I researched WebGPU more, I came across another project, [WebLLM](https://github.com/mlc-ai/web-llm), that supports more LLMs with huge parameters and is working on my machine.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

One way to tell that the WebGPU is running locally is that the 2.7b parameters LLM is consuming almost all of my Nvidia GPU’s memory. The WebGPU API marks a significant improvement in the accessibility and scalability of AI technology, making it more accessible on mobile devices.

This is how the Llama 2.7B model responds to the prompt to create quizzes:

A screenshot of a chat

Description automatically generated A screenshot of a chat

Description automatically generated

# Appendix A: Nemobot Code Reference

The following is the code for the Nemobot and LLM functions. It can be loaded in [New Playground](https://nemobot-neue-experiment.vercel.app/playground/new).

GitHub Repository: <https://github.com/xeroxis-xs/NTU-SC4052-Cloud-Computing>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a chat

Description automatically generated

A screenshot of a chat

Description automatically generated