Models Developed:

Model 1: Sentence to question generator - The input will be text sentences in subjects like English, Physics, Chemistry, and Biology.

Model 2: Mathematics question generator - The input will be the category of Mathematics such as “Integers” and “Lines and Angles”.

Input JSON format:

The input to the model will be given as a single JSON object like below with the given keys.

{"model": "0", "text": "input sentence", "MCQ": "2", "Single_word": "3", "Fill_blank": "4", "Descriptive": "2", "Difficulty": "Medium"}

Where;

model = 0 for choosing English and Science sentences model
model = 1 for choosing Mathematics model
text = The input sentence in case of English and Science sentences model.
text = The category of Mathematics such as “Integers” in the case of Mathematics model.
MCQ = Number of Multiple Choice Questions to be generated.
Single_word = Number of Single word answer questions to be generated.
Fill_blank = Number of fill in the blank type questions to be generated.
Descriptive = Number of Descriptive type questions to be generated.
Difficulty = “Easy” or “Intermediate” or “Advanced”.

Output JSON format:

The model output will also be in JSON format. The two models will have two different JSON formats because the Mathematical questions will also have “Steps to solve”.

Output JSON format of English and Science questions:

{
“multiple_choice_questions” : {“question_number”: “the question number”, “question” : “the question”, “option1” : “first option”, “option2” : “second option”, “option3” : “third option”, “option4” : “fourth option”, “right_answer_number” : “correct option”, “right_answer”: “correct answer”},

“single_word_answer_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”}

“fill_in_the_blank_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”}

“descriptive_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”}

}

Output JSON format of Mathematics questions:

{
“multiple_choice_questions” : {“question_number”: “the question number”, “question” : “the question”, “option1” : “first option”, “option2” : “second option”, “option3” : “third option”, “option4” : “fourth option”, “right_answer_number” : “correct option”, “right_answer”: “correct answer”, “steps_to_solve” : “steps or explanation”},

“single_word_answer_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”, “steps_to_solve” : “steps or explanation”}

“fill_in_the_blank_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”, “steps_to_solve” : “steps or explanation”}

“descriptive_type_questions” : {“question_number”: “the question number”, “question” : “the question”, “right_answer”: “correct answer”, “steps_to_solve” : “steps or explanation”}

}

How to call the output JSON file with keys?

The generated question, answer options, right answer, and steps to solve can be extracted from the output JSON using keys as shown below.

```
“output” will be the model output.

output['multiple_choice_questions'][0]['question_number']  -  To  get  the
first question in “multiple choice questions

output['multiple_choice_questions'][0]['question']  -  To  get  the  first
question in “multiple choice questions

To get the options
output['multiple_choice_questions'][0]['option1']
output['multiple_choice_questions'][0]['option2']
output['multiple_choice_questions'][0]['option3']
output['multiple_choice_questions'][0]['option4']

To get the right answer number
output['multiple_choice_questions'][0]['right_answer_number']


To get the right answer
output['multiple_choice_questions'][0]['right_answer']


To get the Steps to solve/ Explanation
output['multiple_choice_questions'][0]['steps_to_solve']
```

Details of files:

1. Experiments.ipynb - All the models and methods experimented and developed throughout the project and their performance details.
2. Test.ipynb - We can test the models in Google Colab by opening and running this file.
3. Question generator project details.docx - Details of the project.
4. Indbytes_Qgen_Test_Report.pdf - Report of the final testing conducted.
5. lambda_Version_1_python3.11.py
   lambda_Version_2_python3.11.py

lambda_Version_3_python3.11.py
lambda_Version_4_python3.11.py
All the versions of lambda functions. Version_4 is the latest.

6. gpt_python3.8.zip - The customized chatgpt configuration in python 3.8.
7. gpt_python3.11.zip - The customized chatgpt configuration in python 3.11.

How to test in Google Colab?

The models can be tested in Google colab by opening and running the Test.ipynb file. Detailed instructions are given in the file.

How to test in AWS lambda?

The models are also deployed in AWS as a lambda function. It can either be tested using Postman or in the AWS itself by running a test event.

How to check open AI server status?

The real-time status of the server can be checked here - https://status.openai.com/

How to monitor and set limits in API calls?

ChatGPT cost details:
https://hackernoon.com/open-ais-chatgpt-pricing-explained-how-much-does-it-cost-to-use-gpt-models
It will be 0.0004 USD for 1000 tokens. Tokens are words that it generates. We can set the monthly limit in the API also.

The API call limits can be set in the open AI website once logged in.