Nama : Kean Nafis Santang
NIM : 18221148

Welcome to the Critiquill Microservice!

Your one-stop solution for storing debate speeches and generating feedback.

What is Critiquill?

Critiquill is a microservice that allows you to summarize a debate speech into a pre-determined format and **automatically generate feedback for that speech**. The generated feedback form is made using an algorithm that evaluates the overall quality of the speech and by prioritizing the most key aspects. Critiquill also writes both the speech and the feedback generated from it into a JSON file. The feedback generated by the service will grade every aspect of the submitted speech and provide feedback that prioritizes the most important aspect(s) that are still lacking.

How does it work?

Generating Feedback (Main Flow)

1. Before you can get feedback on your debate speech, you need to submit your debate speech to Critiquill for storage. To make the process efficient, Critiquill has provided tutors with a standardized format for all of your speech transcripts. Please transform every speech transcript into the following format
    a. Speech_id : an **integer** that represents the unique speech id
    b. Student_id : a unique **integer** that represents the student that made the speech
    c. Student_name : a **string** that represents the non-unique name of the student who made the speech
    d. Speech_topic : a **string** that describes (in one or two words) the topic of the debate speech
    e. Speech_date : a **string** (converted from a datetime format) that represents the year, month, date, and time when the speech was made
    f. Argument_1 : a **string** that contains the key points of the first argument made by the student
    g. Argument_1_quality : an **integer**, scaled from 1-5, that represents the quality of the first argument (determined by the tutor)
    h. Argument_2 : a **string** that contains the key points of the second argument made by the student
    i. Argument_2_quality : an **integer**, scaled from 1-5, that represents the quality of the second argument (determined by the tutor)
    j. Substance_1 : a **string** that details the first argument made by the student

k. Substance_1_quality : an **integer,** scaled from 1-5, that represents the quality of the substance brought in the first argument
l. Substance_2 : a **string** that details the second argument made by the student
m. Substance_2_quality : an **integer,** scaled from 1-5, that represents the quality of the substance brought in the second argument
n. Structure_type : a **string** that represents the type of structure used by the student in the speech (unstructured/semi-structured/structured)
o. Structure_quality : an **integer,** scaled from 1-5 that represents the quality of the structure of the speech
p. Mannerism_confidence: an **integer,** scaled from 1-5, that represents the level of confidence brought by the student in their speech
q. Mannerism_voice : an **integer,** scaled from 1-5, that represents the quality of voice volume, tempo, and tones brought by the student in their speech

Example:


```
"speech_id": 1,
"student_id": 12,
"student_name": "Kean",
"speech_topic": "Philosophy",
"speech_date": "2023-06-17 00:00:00",
"argument_1": "Bubur sebaiknya tidak diaduk",
"argument_1_quality": 2,
"argument_2": "Es teh manis gabisa anget",
"argument_2_quality": 5,
"substance_1": "Kalau diaduk kerupuknya jadi basah",
"substance_1_quality": 4,
"substance_2": "Kalo anget namanya bukan es",
"substance_2_quality": 3,
"structure_type": "structured",
"structure_quality": 4,
"mannerism_confidence": 3,
"mannerism_voice": 2
```


2. Once you have filled the speech format, use the POST method on the /speeches path to add the speech into the JSON file. Every speech must have a unique speech_id for the response to be valid.

Example:

```
curl -X 'POST' \
  'http://20.92.28.205/speeches' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "speech_id": 4,
  "student_id": 21,
  "student_name": "Santang",
  "speech_topic": "Indonesian Politics",
  "speech_date": "2023-02-02 00:00:00",
  "argument_1": "Lorem ipsum",
  "argument_1_quality": 3,
  "argument_2": "Lorem ipsum",
  "argument_2_quality": 3,
  "substance_1": "Lorem ipsum",
  "substance_1_quality": 3,
  "substance_2": "Lorem ipsum",
  "substance_2_quality": 3,
  "structure_type": "Lorem ipsum",
  "structure_quality": 3,
  "mannerism_confidence": 3,
  "mannerism_voice": 3
}'
```

3. Once you have submitted a speech into the system (or if the speech you want to generate feedback on already exists), use the POST method in the /feedback path to generate feedback for a specific speech. Every speech can only have one feedback, so the program will not work if the speech you want to generate feedback for already has an existing feedback form. When using the method, you will be asked to fill in the following format:

```
{
  "speech_id": 0,
  "student_id": 0,
  "score": 0,
  "area_of_improvement_1": "string",
  "overall_feedback": "string"
}
```

YOU ONLY NEED TO FILL THE speech_id **VARIABLE**
Everything else can be left as is, the system will calculate and process everything else

4. Once you have successfully done step 3, a new feedback form will be generated automatially in the JSON file, each istance of a feedback form will follow the same format:
    a. Speech_id : an **integer** that represents which speech this feedback was generated for
    b. Student_id : an **integer** that represents the student that made the speech
    c. Score : an **integer** that represents the total accumulated score the student received from his/her speech
    d. area_of_improvement_1 : the first aspect that the student could improve on in their speech
    e. area_of_improvement_1 : the second aspect that the student could improve on in their speech
    f. Overall_feedback : the general feedback regarding the performance of the student when delivering his/her speech

    Example:


```
  {
    "speech_id": 3,
    "student_id": 15,
    "score": 25,
    "area_of_improvement_1": "Work on analyzing and improving your first argumen
t",
    "overall_feedback": "Great Speech! You have solid fundamentals and good execu
tion. Work on taking it to the next level",
    "area_of_improvement_2": "Work on adding weight adding substance to your seco
nd argument"
  },
```


5. To read a specific feedback form, use the GET method on the /feedback/{speech_id} path to read an existing feedback based on the speech_id you inserted as the parameter
   Example:

```
curl -X 'GET' \
  'http://20.92.28.205/feedback/{speech_id}}?speechID=2' \
  -H 'accept: application/json'
```

Individual Functions

GET /

The home page that the user first sees when accessing this microservice

GET /speeches

Returns all the speeches that have been stored in the Critiquill JSON file

PUT /speeches

Updates the information in an existing speech form. Uses the same format as every speech form

POST /speeches

Adds a new speech and inserts it into the JSON file

GET /speeches/students/{student_id}

Returns an array of speech_id that contains the speech_ids of all speeches made by the specified student

GET /speeches/{speech_id}

Returns a specific speech form based on the inputted speech_id

DELETE /speeches/{speech_id}

Deletes the speech form in the JSON file that matches the speech_id

GET /feedback

Returns every feedback form stored in the JSON file

PUT /feedback

Updates an existing feedback form. Uses the same format as every feedback form

POST /feedback

Generates a new feedback form based on a specific speech_id and adds it into the JSON file

GET /feedback/{speech_id}

Returns a specific feedback form that matches the inputted speech_id

DELETE /feedback/{speech_id}

Deletes the feedback form in the JSON file that matches the speech_id