OmegaUp
Ai Teaching Assistant
Lorem ipsum dolor elt amet, conectetuyrylelur
sed diepolina tidlysmod lpnet luquss

AI Teaching Assistant

Nandini

Nandini Jaiswal

nandinijaiswal2002

Technical skills

Nandini

I completed my B.Sc. in Statistics from Bethune College, affiliated with Calcutta University (July 2023), and joined CMI to pursue an M.Sc. in Data Science (2023 - 2025). In my B.Sc., I secured 3rd position in statistics at Calcutta University with a CGPA of 9.037.

This summer I interned at Coriolis Technologies, where I got a chance to explore some machine learning models, some LLMs, and various deep learning algorithms, which provided me with some in-depth understanding and practical experience. My work was mostly focused on computer vision and generative AI. I had to automate the web UI testing process by leveraging ML models and AI. This can be considered an alternative to Cypress and Selenium, but it is so much better and easier to use than these tools. Besides, it requires no prior knowledge of coding.

Moreover, I am exploring deep learning (natural language processing, transformers, CNNs, and RNNs), which gives me a better understanding of industry algorithms. I am also studying predictive analysis, time series analysis, and applied data analytics, which provide me with a great hands-on experience of working and playing with data. I have studied Big Data Analytics, where I gathered some insight into Hadoop, Neo4j, MapReduce, and Pig. Apart from these, I have in-depth knowledge of using SQL and PowerBI. I intend to do some guided projects and industry projects in the same field in my upcoming semester. As a part of my coursework, I have done a lot of assignments using ML and deep learning models. I had a CGPA of 8.67 until the third semester.

Here is the link to omegaUp merged PRs**.**

Motivation

The project is of immense importance to OmegaUp. Upon successful implementation, this project will help teachers give AI feedback in no time. Let me pose the scenario:

- A teacher creates a course on OmegaUp and uploads an assignment that contains a single question.
- Many students join the course and then try to solve the question by submitting their codes.
- Most of them pass. Some of them tried once; a few of them tried twice or thrice, but they are facing some trouble understanding why her/his code is wrong.
- They ask for feedback, and now, there's a ton of code that the teacher might go through.
- Even if they don't ask, we should try to proactively detect when a student is stuck and provide them with help even if they don't explicitly ask for it
- With the help of the AI TA, this seems to be a lot easier for the teacher. They can definitely catch silly mistakes. They can explain complex concepts in brief. They can refer to good study materials. So, initial feedback from the TA might resolve a large number of queries. For follow-up requests, it's on the teacher to decide whether she/he would use AI TA or go through the code her/himself to resolve the queries.
- Moreover, the teacher can go through the AI's response before posting it. (She/he can also choose to skip this confirmation and give AI TA the whole power.)

Short description

The aim is to create an "AI Teaching Assistant" bot that answers the relevant queries and helps teachers and students with code reviews and answer clarifications.

Detailed description

Let's dive deeper into the details of the project.
First of all, the initial version of the project prototype is already implemented and can be tracked here.
As the bot works both as an AI teaching assistant and code reviewer, it must handle all types of queries and simultaneously must not be misused by the students.
The type of queries it would handle should cover the following:

- Explain the topic "binary search."
- Explain the question. (In that case, the bot should not return the code solution and should return only the detailed explanation.).

- May you help me with the hints for this question? (Again a hard query because we need to make sure that no code solution is returned.)
- Why is my solution wrong? Is the error logical or syntactical?

The bot receives these queries and uses a curated prompt that fetches responses from the LLM. Then, the fetched response is parsed and posted as feedback. (Refer to the workflow plan section below.)

Key objectives

- We should incorporate instantaneous feedback processing, i.e., if a teacher enables AI TA for a course, any student asking for feedback will get instant feedback.
- Though the AI TA works fine, our main aim for the first phase will be to make it more robust and run it on a few demo courses to assess its accuracy.
- Ideally we should soon be able to integrate the AI TA script with the omegaup platform with the help of RabbitMQ.
- Following that, we have two major tasks:
  - Assess the best among LLMs and make it the default LLM
  - Make the code multithreaded to speed up the whole workflow

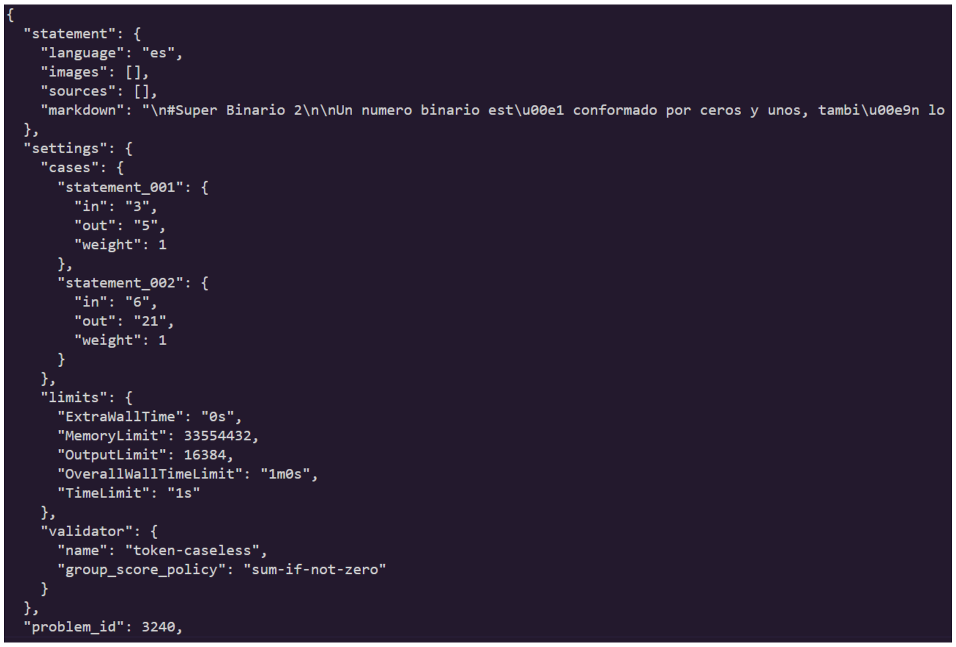
Final plans

(Low priority)
After all this, we need to create a UI that will greatly ease things up for the teachers. The teachers should be able to see the questions, answers, and the feedback requests and even review the feedback before posting the feedback.

API endpoints

The following endpoints are currently used:

- `/api/problem/details/`: will fetch the details of the problems.

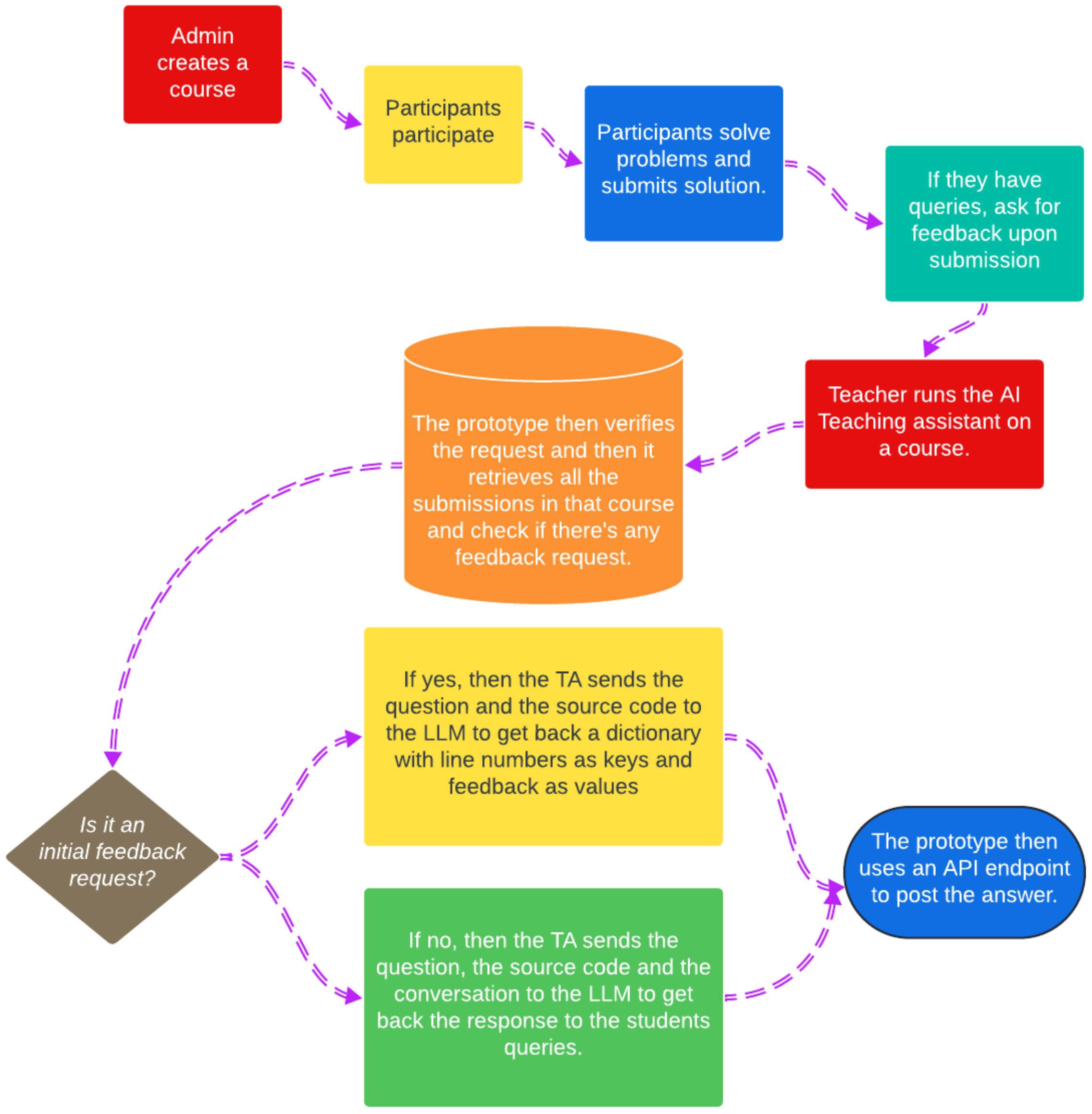


```
{
  "statement": {
    "language": "es",
    "images": [],
    "sources": [],
    "markdown": "\n#Super Binario 2\n\nUn numero binario est\u00e1 conformado por ceros y unos, tambi\u00e9n lo
  },
  "settings": {
    "cases": {
      "statement_001": {
        "in": "3",
        "out": "5",
        "weight": 1
      },
      "statement_002": {
        "in": "6",
        "out": "21",
        "weight": 1
      }
    },
    "limits": {
      "ExtraWallTime": "0s",
      "MemoryLimit": 33554432,
      "OutputLimit": 16384,
      "OverallWallTimeLimit": "1m0s",
      "TimeLimit": "1s"
    },
    "validator": {
      "name": "token-caseless",
      "group_score_policy": "sum-if-not-zero"
    }
  },
  "problem_id": 3240,
```

- `/api/problem/clarifications/`: will fetch the clarifications, if any.
- `/api/problem/solution/`: will fetch the solution of the problem if available.
- `/api/run/details/`: will fetch details about the submitted code and the status.
- `/api/run/getSubmissionFeedback/`: will list all the existing feedback for a submission.
- `/api/submission/setFeedback/`: will post the most recent feedback to the submission.
- `/api/course/runs/`: fetches all the runs from a course.

Current Workflow

- A course is created by the teacher.
- Students participate in the course.
- Students submit some solutions or go through the course content.
- Students may have some doubts about the content or about the question, or he/she may ask for code reviews for the submitted solution.
- The teacher may run this prototype to address all such feedback.
- Then the script will go through the feedback requests and answer them either directly or as code reviews.
- The reviews and the answers will be posted directly to the students using some API endpoints.


Admin creates a course
Participants participate
Participants solve problems and submits solution.
If they have queries, ask for feedback upon submission
Teacher runs the AI Teaching assistant on a course.
The prototype then verifies the request and then it retrieves all the submissions in that course and check if there's any feedback request.
Is it an initial feedback request?
If yes, then the TA sends the question and the source code to the LLM to get back a dictionary with line numbers as keys and feedback as values
If no, then the TA sends the question, the source code and the conversation to the LLM to get back the response to the students queries.
The prototype then uses an API endpoint to post the answer.

Currently, this “chain of thoughts” style of prompting is used.

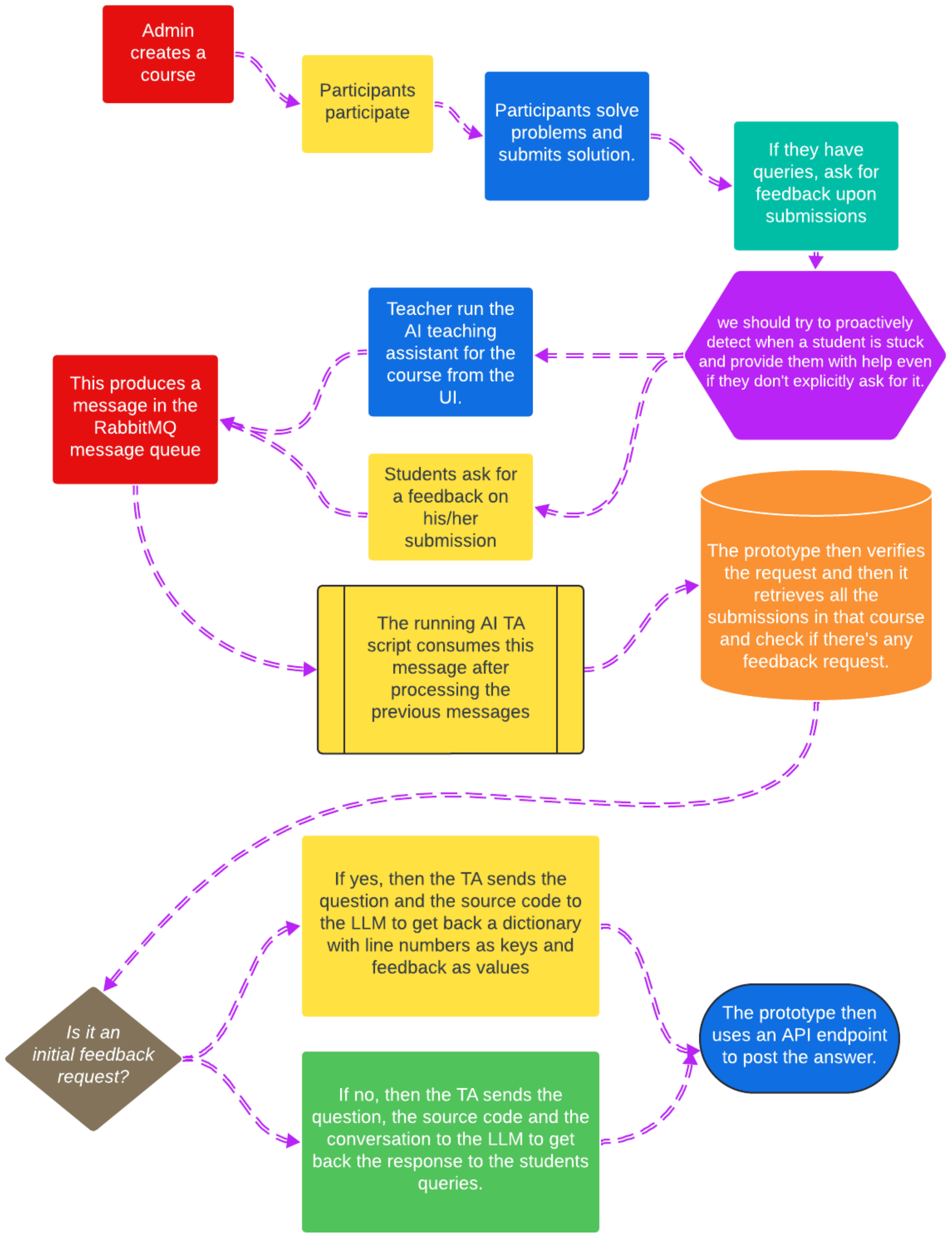
```
You are a teaching assistant, and your goal is to help students
with specific programming-related queries without directly
providing full solutions. Follow these steps to guide users
based on their query type: \
1) When a student asks for a topic explanation (for example,
\"Binary Search\"), provide a detailed breakdown of the concept
without solving any specific problems. \
```

2) If a student asks for a question explanation, ensure that you describe the problem's details, clarifying requirements, constraints, and logic without offering any code. \
3) If the student requests hints for a problem, give guidance on approaching the problem (example, breaking it down, algorithms to consider) without revealing the final code. \
4) When asked why a solution is wrong, do the following: First, analyze the student's solution and determine if it is on the right track or completely off. If it's off-track, gently point out that the approach needs reconsideration. If the solution is on the right track, identify the approach the student has taken (example, brute force, two-pointers, hash table, etc.). If the approach is inefficient or incorrect (example, brute force for large inputs), suggest that the student consider more optimal techniques. \
5) Carefully examine the code for syntax errors and provide specific feedback on those issues. \
6) If you find any logical error in the code, gently point out the mistake but do not give the solution for the mistake. \
7) When asked if a solution is correct or not, do not answer. \
8) If a student is getting a wrong answer for some general mistake (for example, not leaving space between two numbers, asking for input by typing a message instead of a standard input, etc.), do point that out. \
9) If you see any irrelevant print statement, ask the student to comment out that particular statement. \
10) Before giving any remarks or responses, solve the problem on your own and then compare your solution to the student's solution. \
11) If asked to explain the solution, give one or two hints or explain the logic that can help students arrive at the correct solution. \
12) Remember, your goal is to facilitate the teaching process and not to provide the solution directly. \

```
13) Keep your message clear and concise in less than 150 to 200
words. \
14) If a code snippet is submitted, return the answer in the
json format only.The line number (0-indexed) of the feedback
should be the key, general advices should be under 'general
advices' key. \
15) If only a general question is asked and no code snippet is
submitted, return the output in normal text format (not in json
format). \
16) Please return the response in {LANGUAGE}.\
17) Keeping all those in mind, please answer the following
query: {query_content}.
```

Final Workflow Plan

We plan to have this workflow on the user level:

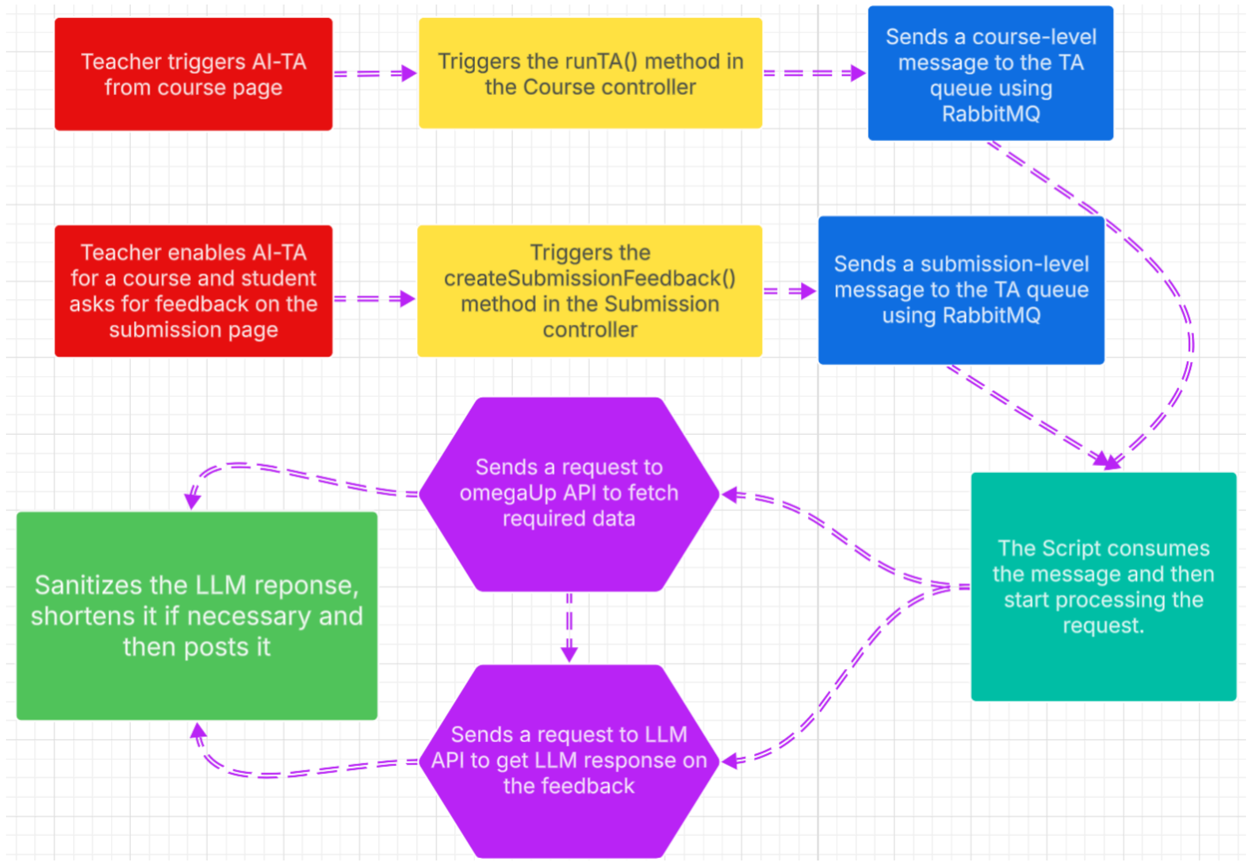

Admin creates a course
Participants participate
Participants solve problems and submits solution.
If they have queries, ask for feedback upon submissions
we should try to proactively detect when a student is stuck and provide them with help even if they don't explicitly ask for it.
Teacher run the AI teaching assistant for the course from the UI.
Students ask for a feedback on his/her submission
This produces a message in the RabbitMQ message queue
The running AI TA script consumes this message after processing the previous messages
The prototype then verifies the request and then it retrieves all the submissions in that course and check if there's any feedback request.
Is it an initial feedback request?
If yes, then the TA sends the question and the source code to the LLM to get back a dictionary with line numbers as keys and feedback as values
If no, then the TA sends the question, the source code and the conversation to the LLM to get back the response to the students queries.
The prototype then uses an API endpoint to post the answer.

Need to figure out: If the student asks for feedback on his/her wrong submission, should we process that immediately? I would expect that the feedback will be rolled out once the teacher runs the AI TA. Or, if the teacher has run it even once, then we can enable the instantaneous feedback processing.

So, we will have two options to run AI TA.

- The teacher runs the AI TA for all the submissions.
- The teacher enables AI TA, and then when a student asks for feedback, the AI TA will process those requests immediately. (Even if the student doesn't ask, if he/she is stuck for too long, the AI TA should identify and offer feedback.)

Here is a workflow on the component level:

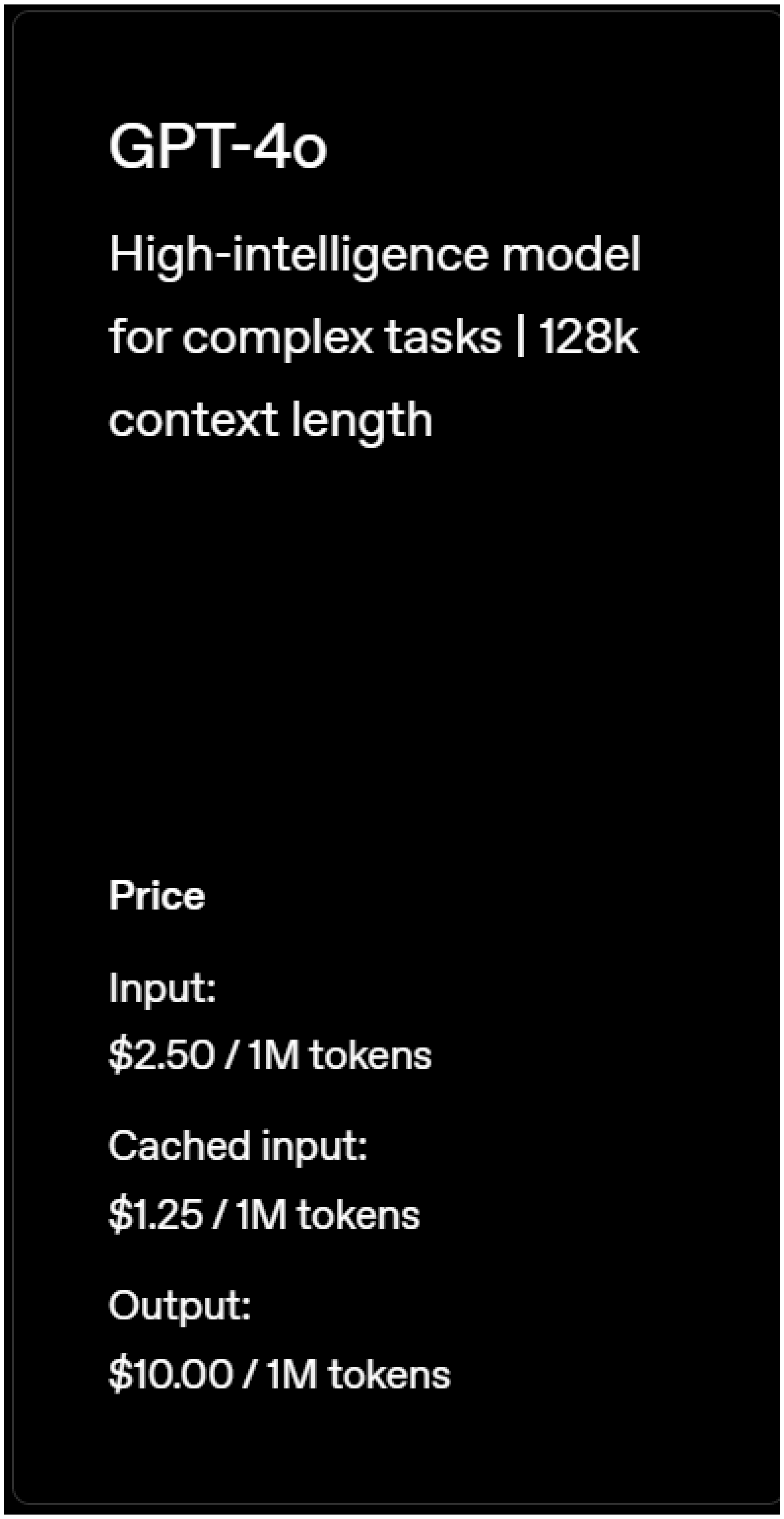

Teacher triggers AI-TA from course page
Triggers the runTA() method in the Course controller
Sends a course-level message to the TA queue using RabbitMQ
Teacher enables AI-TA for a course and student asks for feedback on the submission page
Triggers the createSubmissionFeedback() method in the Submission controller
Sends a submission-level message to the TA queue using RabbitMQ
Sends a request to omegaUp API to fetch required data
The Script consumes the message and then start processing the request.
Sanitizes the LLM reponse, shortens it if necessary and then posts it
Sends a request to LLM API to get LLM response on the feedback

Exploring LLMs

Currently, we use the **GPT-4o** model as the LLM. We have used **GPT-3.5**, **GPT-4o**, **GPT-4o mini,** and a few other models, and **the GPT-4o** model outperformed others.

GPT-4o

GPT-4o is a fairly accurate model with a pretty low cost. The reasoning of answers is good, as well as the fact that it can extract the line numbers with good precision.



GPT-4o

High-intelligence model for complex tasks | 128k context length

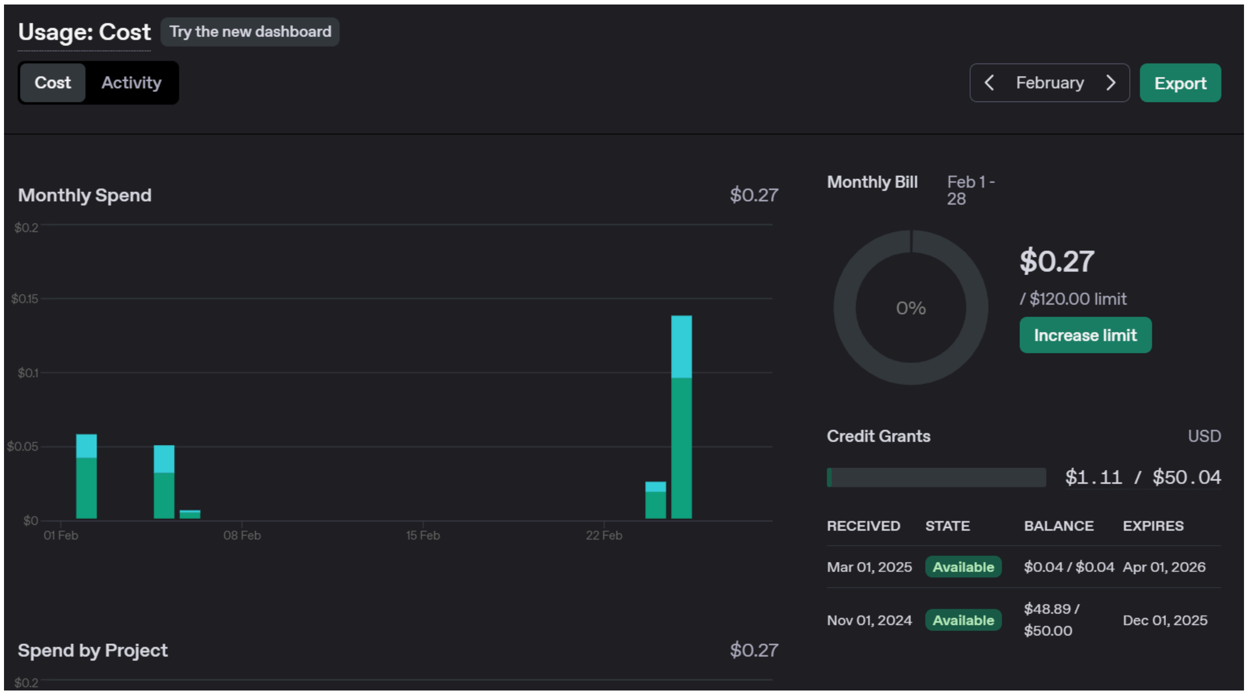
Price

Input:
$2.50 / 1M tokens

Cached input:
$1.25 / 1M tokens

Output:
$10.00 / 1M tokens

After processing a huge number of feedbacks in February, we can see that only $0.27 was spent.



Credit Grants USD

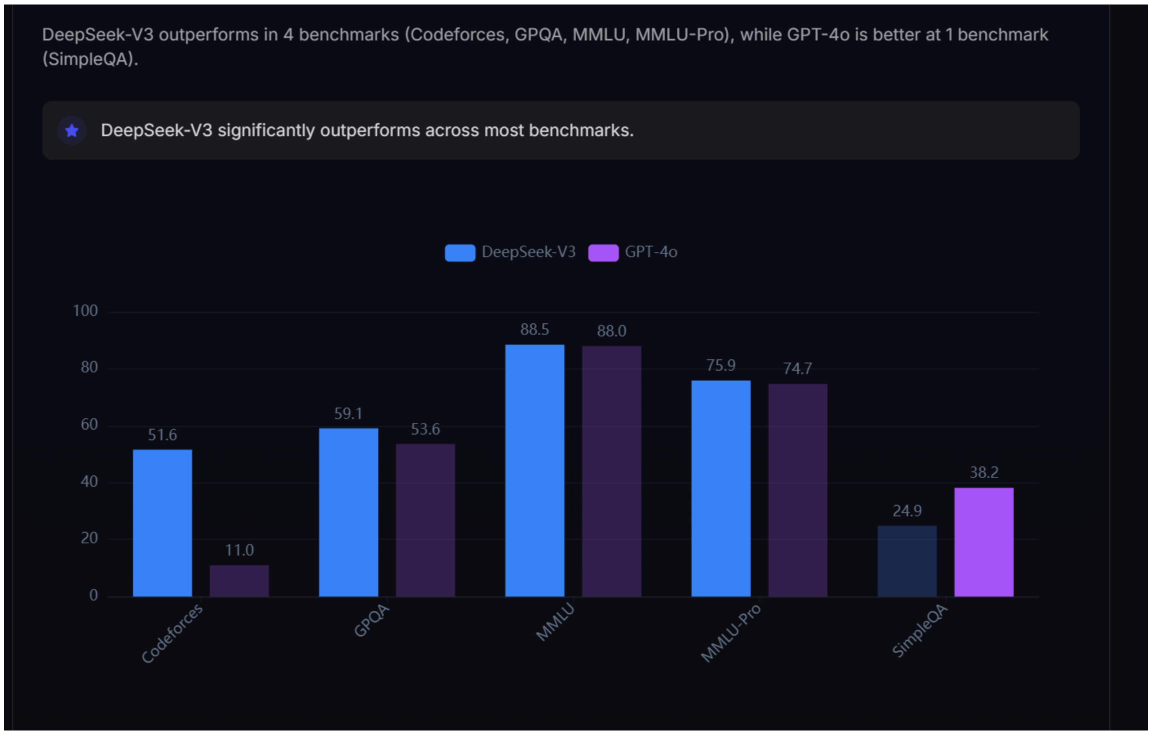
$1.11 / $50.04

RECEIVED	STATE	BALANCE	EXPIRES
Mar 01, 2025	Available	$0.04 / $0.04	Apr 01, 2026
Nov 01, 2024	Available	$48.89 / $50.00	Dec 01, 2025

Deepseek

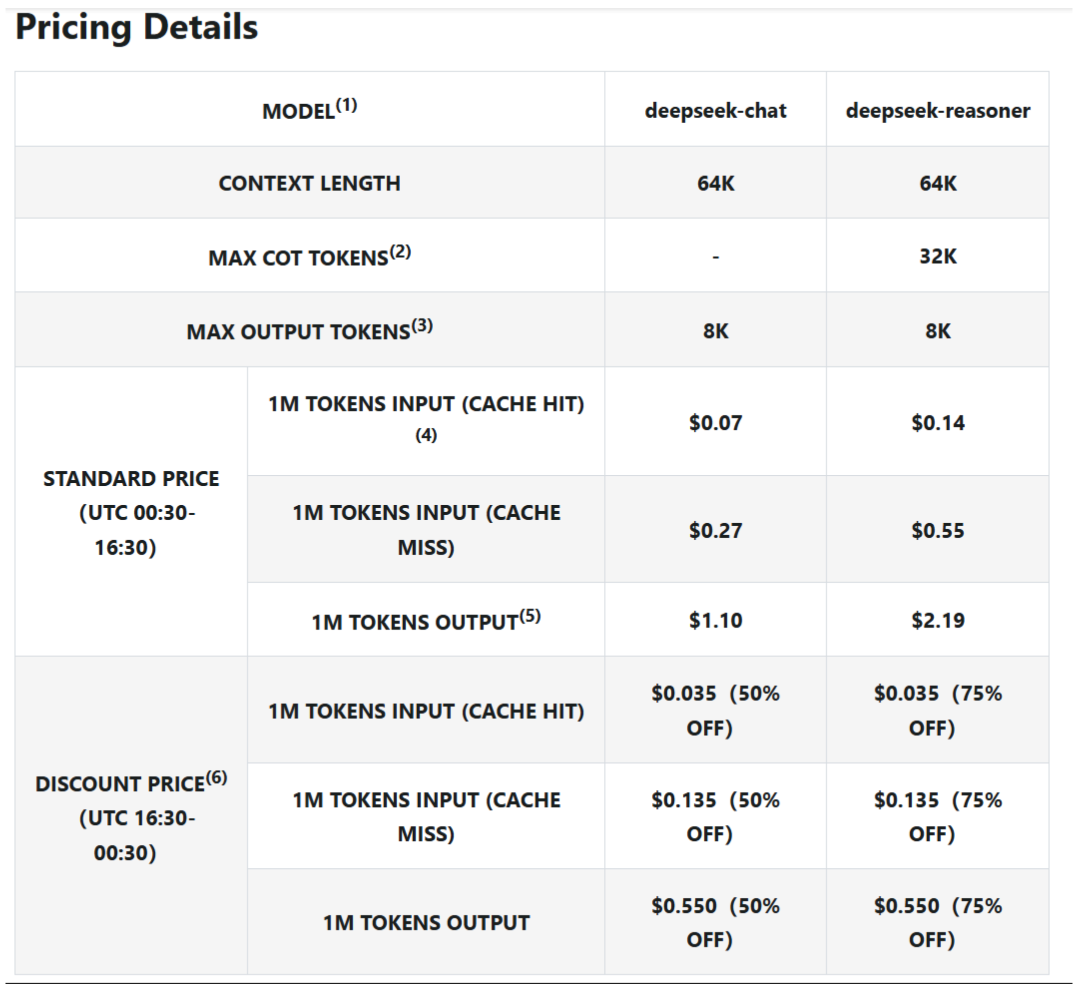
As it turned out, Deepseek seems to be even better.

- The model is more accurate with better reasoning and higher accuracy.


DeepSeek-V3 outperforms in 4 benchmarks (Codeforces, GPQA, MMLU, MMLU-Pro), while GPT-4o is better at 1 benchmark (SimpleQA).

DeepSeek-V3 significantly outperforms across most benchmarks.

- It’s a lot cheaper compared to the model we are using (it even has a discounted price if used at a specific time).
- It’s open-source.



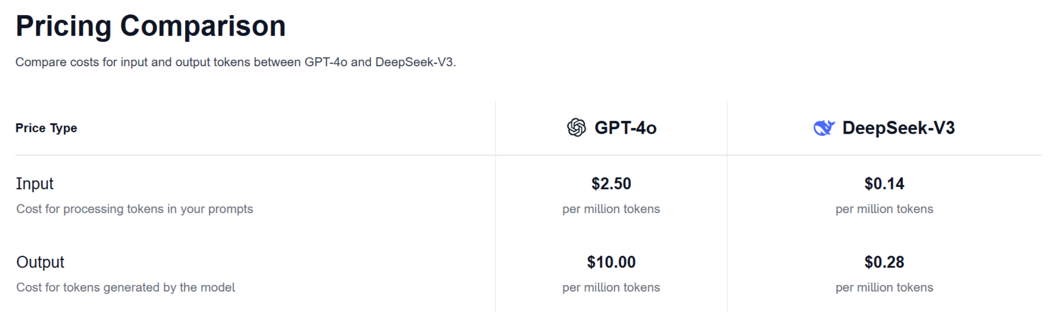
Pricing Details

MODEL[1]		deepseek-chat	deepseek-reasoner
CONTEXT LENGTH		64K	64K
MAX COT TOKENS[2]		-	32K
MAX OUTPUT TOKENS[3]		8K	8K
STANDARD PRICE (UTC 00:30-16:30)	1M TOKENS INPUT (CACHE HIT)[4]	$0.07	$0.14
	1M TOKENS INPUT (CACHE MISS)	$0.27	$0.55
	1M TOKENS OUTPUT[5]	$1.10	$2.19
DISCOUNT PRICE[6] (UTC 16:30-00:30)	1M TOKENS INPUT (CACHE HIT)	$0.035 (50% OFF)	$0.035 (75% OFF)
	1M TOKENS INPUT (CACHE MISS)	$0.135 (50% OFF)	$0.135 (75% OFF)
	1M TOKENS OUTPUT	$0.550 (50% OFF)	$0.550 (75% OFF)

Another big advantage while shifting from OpenAI to DeepSeek will be that DeepSeek uses OpenAI SDK. We just need to modify the “base_url,” and that’s it. Everything will work just fine.

```python
from openai import OpenAI

client = OpenAI(api_key="<DeepSeek API Key>", base_url="https://api.deepseek.com")

response = client.chat.completions.create(
    model="deepseek-chat",
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Hello"},
    ],
    stream=False
)

print(response.choices[0].message.content)
```

Price comparison



Pricing Comparison

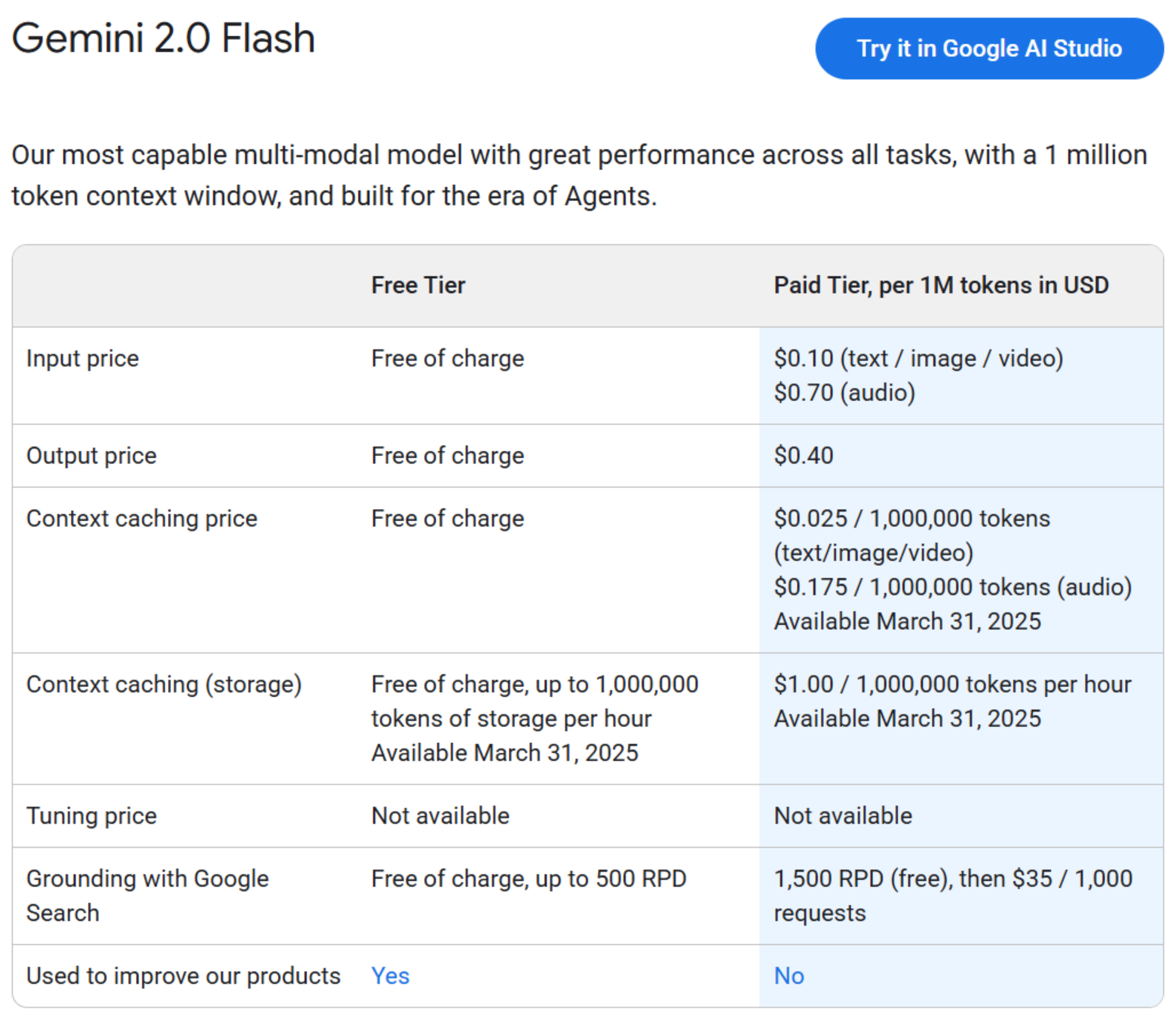
Compare costs for input and output tokens between GPT-4o and DeepSeek-V3.

Price Type	GPT-4o	DeepSeek-V3
Input Cost for processing tokens in your prompts	**$2.50** per million tokens	**$0.14** per million tokens
Output Cost for tokens generated by the model	**$10.00** per million tokens	**$0.28** per million tokens

Claude and Gemini

Apart from that, we have two more models.

- **Claude Sonnet**: It’s extremely good when it comes to reasoning, but its cost is comparable to GPT-4o.
- **Gemini**: I have never explored the performance of the model in the context of coding, but it has free tiers for many of its models. (Refer to this reddit blog which says that the free tiers use our data to train.)



Gemini 2.0 Flash

Try it in Google AI Studio

Our most capable multi-modal model with great performance across all tasks, with a 1 million token context window, and built for the era of Agents.

	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Free of charge	$0.10 (text / image / video) $0.70 (audio)
Output price	Free of charge	$0.40
Context caching price	Free of charge	$0.025 / 1,000,000 tokens (text/image/video) $0.175 / 1,000,000 tokens (audio) Available March 31, 2025
Context caching (storage)	Free of charge, up to 1,000,000 tokens of storage per hour Available March 31, 2025	$1.00 / 1,000,000 tokens per hour Available March 31, 2025
Tuning price	Not available	Not available
Grounding with Google Search	Free of charge, up to 500 RPD	1,500 RPD (free), then $35 / 1,000 requests
Used to improve our products	Yes	No

Here is the list of a few open LLMs.

Planned Enhancements

- ### Implement Processing Single Submission

  Currently, the AI TA processes all the submissions in a course and then gives feedback. However, as we are planning on on-the-fly feedback, we should write a very similar code that takes a single submission and offers feedback.

- ### Adding tests

  I plan to add the tests for the following APIs.
  - Omegaup API
  - LLM API

For both of them, I will add unit tests using pytest.

- ## Handling errors gracefully

  Currently, the code is not robust; a single failure can stop the whole process, which takes a considerable amount of time. So, it's a good idea to enable meaningful error messages and handle errors to keep the code running if the encountered error is skippable.

- ## Making the "assignment alias" optional

  Given a course alias, we would ideally expect the TA to provide feedback on all the assignments (if not specifically given). However, the current implementation must have an assignment alias to proceed.

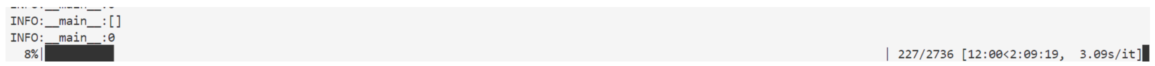
- ## Intelligently detecting the struggling students

  The current implementation gives feedback only if the students ask for the feedback. We want to integrate a feature that detects if a student is struggling with the problem by looking at his/her last few submissions and automatically gives feedback if positive.
  We would need to process this on the fly. One simple logic will be to extract all the submissions of a user for a particular problem. If this exceeds a threshold, we add that submission to the message queue. However, more complex and robust logic can be implemented after observing some patterns.

- ## Multiprocessing

  While this is not currently planned, once we integrate the AI TA to the omegaup domain, multiprocessing might speed up the process by around 10x, which will be great as the TA might need thousands of feedbacks to process.

- ## Improving time complexity

  ```
  INFO:__main__:[]
  INFO:__main__:0
    8%|████                                                  | 227/2736 [12:00<2:09:19,  3.09s/it]
  ```

  It's currently taking 2 hours to process 2736 submissions, of which hardly 30 have requested feedback. What we can do is add a parameter to the API endpoint to only return submissions with feedback requests.

- ## Adding verdict

  Sometimes, the TA makes some general feedback, which is kind of ambiguous, as the TA doesn't know the verdict of the answer. Adding a verdict to the feedback request might increase the accuracy.
  We can add the following prompts for each verdict.

```
{
```

```
        WA: 0, // Extract the logic from the code and see what’s
wrong.
        PA: 0, // What optimizations can be done to pass all the
test cases?
        AC: 0, // Congratulate. Comment alternative approaches (if
relevant)
        TLE: 0, // What’s causing the time limit to exceed?
        MLE: 0, // Where can the memory be optimized?
        OLE: 0, // Is there an infinite loop or extra print
statemen?
        RTE: 0, // Is there a division by zero or an array out of
bounds?
        CE: 0, // Where is the compilation error?
        JE: 0,
        VE: 0,
      },
```

Database

We need to add a field, “**enable_teaching_assistant**,**”** to the **course table** in the database. We also need to make required changes in **Course DAO** and **VO**.

```
class Courses extends \OmegaUp\DAO\VO\VO {
    const FIELD_NAMES = [
        'course_id' => true,
        'name' => true,
        'description' => true,
        'objective' => true,
        'alias' => true,
        'group_id' => true,
        'acl_id' => true,
        'level' => true,
        'start_time' => true,
        'finish_time' => true,
        'admission_mode' => true,
        'school_id' => true,
        'needs_basic_information' => true,
        'requests_user_information' => true,
        'show_scoreboard' => true,
        'languages' => true,
        'archived' => true,
        'minimum_progress_for_certificate' => true,
        'certificates_status' => true,
        'recommended' => true,
        'enable_teaching_assistant' => true, // Add this
    ];
}
```

Instead of enabling `enable_teaching_assistant` manually, we can also set it to `true` when the TA is run on the course for the first time.

Once the `enable_teaching_assistant` is `true`, every time the student asks for feedback, the RabbitMQ will automatically handle the script to run the TA on that particular feedback.

Backend

We are creating a standalone Python script, which is calling the APIs and then processing the requests.
So, we don't need to directly add any **controllers** for the first half at least. When we create a new API endpoint for the AI TA (which is a low priority), we will need to work on creating controllers.

To aid the teaching assistant, we had to and might need to tweak a few endpoints in a few controllers and DAOs.

One of them will be to modify the Runs DAO to modify the getAllRuns() endpoint.

```
    final public static function getAllRuns(
        ?int $problemsetId,
        ?string $status,
        ?string $verdict,
        ?int $problemId,
        ?string $language,
        ?int $identityId,
        ?int $offset = 0,
        ?int $rowCount = 100,
        ?string $execution = null,
        ?string $output = null,
        ?bool $withFeedback = False // Add this
    ):
```

This will help us extract only the runs with feedback, which should greatly speed up the process.

After this, under the **course controller**, we need to create an endpoint named *enableTA()* or *toggleTA()* that, upon request, flips the value of `enable_teaching_assistant`.

Then, we need to handle two types of requests.

- When the teacher runs the AI-TA from the course page, the following process will happen.
    - First, we check if the `enable_teaching_assistant` is set to True. Otherwise, we set it to True.
    - We need to implement a RabbitMQ connection under a function runTA() in the **course controller** as shown. Whenever the user runs the AI-TA, it will set up a connection with RabbitMQ and pass the message needed for TA Script.

```
// Need to add
public static function runTA()
         // set RabbitMQ client parameters
        $routingKey = 'TAQueue';
        $exchange = 'feedbacks';

        // connection to rabbitmq
        $channel =
\OmegaUp\RabbitMQConnection::getInstance()->channel();

        // prepare the message
        $messageArray = [
            'username' => (username),
            'password' => (password),
            'course_alias' => (course_alias),
            …
        ];
        $messageJSON = json_encode($messageArray);
        $message = new
\PhpAmqpLib\Message\AMQPMessage($messageJSON);

        // send the message to RabbitMQ
        $channel->basic_publish($message, $exchange, $routingKey);
        $channel->close();
```

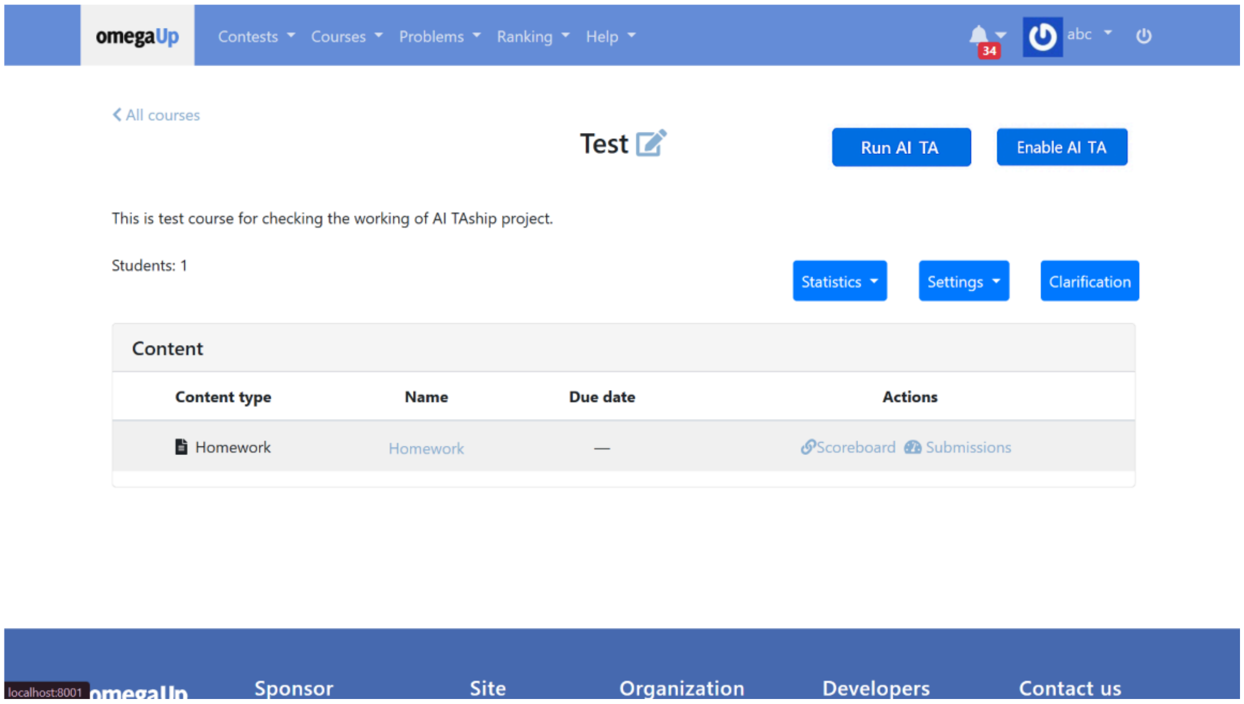
- When the student asks for feedback, the following process will happen.
    - The code first checks if the AI-TA is enabled for the course or if the submission is for any course at all. If not, skip.
    - Similarly, we need to set up a RabbitMQ Connection in the **Submission Controller**. Upon requesting feedback, this will send a message to the MQ needed for the script.

```
public static function createOrUpdateFeedback()
        // Need to add
        // set RabbitMQ client parameters
        $routingKey = 'TAQueue';
        $exchange = 'feedbacks';

        // connection to rabbitmq
        $channel =
\OmegaUp\RabbitMQConnection::getInstance()->channel();

         // prepare the message
        $messageArray = [
            'username' => (username),
            'password' => (password),
            'guid' => (guid),
            …
        ];
        $messageJSON = json_encode($messageArray);
        $message = new
\PhpAmqpLib\Message\AMQPMessage($messageJSON);

        // send the message to RabbitMQ
        $channel->basic_publish($message, $exchange, $routingKey);
        $channel->close();
```

Note that, first of all, we need to modify the script to handle both kinds of requests (mentioned in detail under **Planned Enhancement)**.

Incorporating to omegaup UI (Frontend)

Initially, we would need to add an option for enabling the AI TA from the course page.
Once this button is clicked, we will change the `` `enable_teaching_assistant` `` to True, and this button will be changed to **Disable AI TA**.



This is test course for checking the working of AI TAship project.

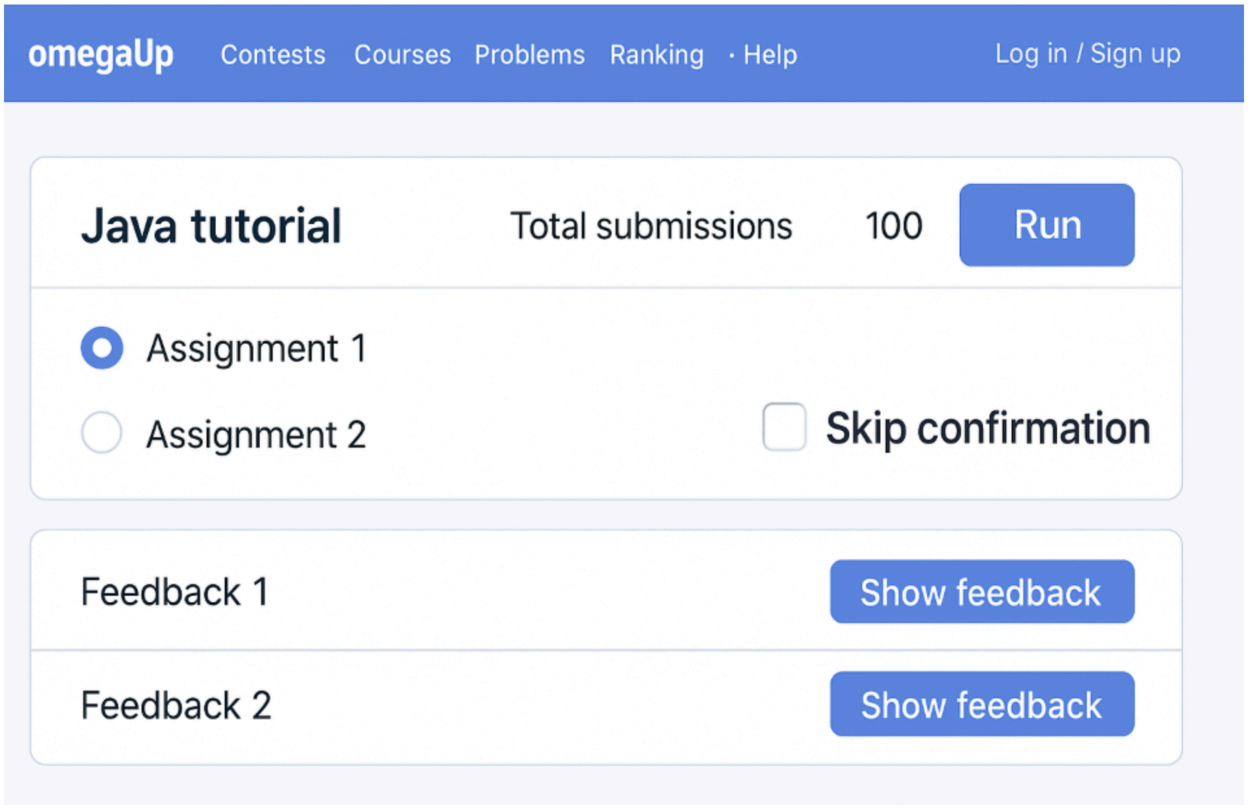
Students: 1

Content

Content type	Name	Due date	Actions
Homework	Homework	—	Scoreboard Submissions

(**Low-priority**)
After the script is ready and the message queue is set up, we intend to create a UI for better UX. What I plan is to create a new endpoint (/teaching_assistant) that will be a standalone page for reviewing feedback using AI.
The teachers will see all the courses they are one of the teachers/owners of. For each course, they will have options to run the AI TA on the course level and on the assignment level. The teacher will also have the option to review the feedback before posting.
After the reviews are posted, we will show a list of already done reviews for each submission with pagination, sorts, and filters.

A very crude design (sorry :3).



Security impact

Whatever we have done till now and whatever I have planned in this document, don't directly modify the database. Every time we access and modify a database using APIs, so APIs being secure implies the current implementation should be secure in that sense.
However, we absolutely need to monitor that the TA we created can't be misused by the students and it doesn't give an unfair advantage to some students.

Deployment Plan

We would run the Python script continuously, and it would consume messages from a message queue. We would like to implement a "RabbitMQ" message queue.

Note that we have two kinds of requests, which the script handles:

- Running the AI-TA for a whole course, requested by a teacher.
- Asking for feedback for a single submission, triggered by a student.

```
try:
       with rabbitmq_connection.connect(
               username=args.rabbitmq_username,
               password=args.rabbitmq_password,
               host=args.rabbitmq_host,
               for_testing=args.test
       ) as channel:
           callback = teaching_assistant_callback.TACallback(
               for_testing=args.test
           )
           rabbitmq_client.receive_messages(queue='teaching_assistant',
                                            exchange='feedbacks',
                                            routing_key='TAQueue',
                                            channel=channel,
                                            callback=callback)
```

Corresponding to this, we will create a class, the TAcallback class, with a __call__() method inside it that will consume a message from the queue and process it, i.e., process the feedback.

Alternatives considered

While writing the initial prototype, we considered a lot of alternatives and chose the current one to be the most plausible implementation. I'm not discussing those alternatives.

However, at this point, one of the few alternatives can be using Kafka as a message queue instead of RabbitMQ. But there's already an implementation of RabbitMQ in our codebase, which will help set up our own.

Another alternative will be to set the TA option on the course page itself so that it's easier to view the course contents before processing the reviews. However, this will be a problem when the teacher wants to run feedback on many courses and view all previous feedback.

Testing plan

Like other components, the AI teaching assistant will have pytest unit and integration tests that we have already discussed above.

Schedule

May 2	Proposal accepted or rejected.	• **Community Bonding Period** - Discussing the project with mentors and finalizing the project implementation.
May 27	Pre-work complete	• **Coding officially begins!**
Jun 6	Milestone #1	• Modify the script to work on a single submission/feedback. • Handle errors gracefully to ensure continuous development
June 13	Milestone #2	• Make the assignment alias argument optional. • Add tests for the TA
June 20	Milestone #3	• Write code to intelligently detect the struggling students. • Improve the time complexity and add verdicts to the feedback.
June 27	Milestone #4	• Add the scope of multiprocessing.
July 8 July 14	Phase 1 Evaluation	• During this period, I need to submit a report to my mentors on the performance of the AI teaching assistant. • Start the process of integrating the TA to the omegaup platform.
July 19 July 26	Milestone #5	• Add/modify endpoints under the course and submission controller to set up RabbitMQ connections.

August 2	Milestone #6	• Implement the RabbitMQ message queue and integrate it with the Python script.
August 9	Milestone #7	• Write codes to implement the UI for the teaching assistant.
August 25 September 1	Milestone #7	• Writing the final blog for the project • Final evaluation starts.

This timeline is tentative and adjustable based on the mentors' feedback, advice, and current progress.