```
from concurrent.futures import ThreadPoolExecutor

import loguru
from dotenv import load_dotenv
from together import Together

load_dotenv()



class TogetherLLM:
    """
        A class to run models with various arguments, including support for concurrent and batch processing.

    Attributes:
        api_key (str): The API key for the model service.
        model_name (str): The name of the model to run.
    """

    def __init__(
        self,
        api_key: str,
        model_name: str,
        system_prompt: str = None,
        *args,
        **kwargs,
```

```
):
    """
    Initializes the ModelRunner with an API key and model name.

    Args:
        api_key (str): The API key for the model service.
        model_name (str): The name of the model to run.
    """
    self.api_key = api_key
    self.model_name = model_name
    self.system_prompt = system_prompt
    self.client = Together(api_key=self.api_key, *args, **kwargs)
    loguru.logger.add("model_runner.log", rotation="10 MB")

def run(self, task: str, *args, **kwargs) -> str:
    """
    Runs the model with the given task and returns the response.

    Args:
        task (str): The task to pass to the model.
        **kwargs: Additional keyword arguments to pass to the model.

    Returns:
        str: The content of the first response choice.
    """
    try:
```

```
        response = self.client.chat.completions.create(
            model=self.model_name,
            messages=[
                {"role": "system", "content": self.system_prompt},
                {"role": "user", "content": task},
            ],
            *args,
            **kwargs,
        )
        loguru.logger.info(
            f"Model {self.model_name} run successfully with task: {task}"
        )
        return response.choices[0].message.content
    except Exception as e:
        loguru.logger.error(
            f"Error running model {self.model_name} with task: {task}: {e}"
        )
        return "Error running model."


def run_concurrently(self, tasks: list, **kwargs) -> list:
    """
    Runs the model concurrently with multiple tasks and returns a list of responses.

    Args:
        tasks (list): A list of tasks to pass to the model.
        **kwargs: Additional keyword arguments to pass to the model.
```

```
    Returns:
        list: A list of responses, each being the content of the first response choice.
    """
    responses = []
    with ThreadPoolExecutor() as executor:
        futures = [
            executor.submit(self.run, task, **kwargs)
            for task in tasks
        ]
        for future in futures:
            try:
                response = future.result()
                responses.append(response)
            except Exception as e:
                loguru.logger.error(
                    f"Error running model concurrently: {e}"
                )
                responses.append("Error running model.")
    return responses


def run_batch(
    self, tasks: list, batch_size: int = 10, **kwargs
) -> list:
    """
    Runs the model in batches with multiple tasks and returns a list of responses.
```

```
        Args:
            tasks (list): A list of tasks to pass to the model.
            batch_size (int): The size of each batch to process concurrently.
            **kwargs: Additional keyword arguments to pass to the model.

        Returns:
            list: A list of responses, each being the content of the first response choice.
        """
        responses = []
        for i in range(0, len(tasks), batch_size):
            batch_tasks = tasks[i : i + batch_size]
            batch_responses = self.run_concurrently(
                batch_tasks, **kwargs
            )
            responses.extend(batch_responses)
        return responses



# # Example usage
# if __name__ == "__main__":
#     model_runner = TogetherLLM(
#         api_key=os.environ.get("TOGETHER_API_KEY"),
#         model_name="meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo",
#         system_prompt="You're Larry fink",
#     )
```

```
#     tasks = [
#         "What are the top-performing mutual funds in the last quarter?",
#         "How do I evaluate the risk of a mutual fund?",
#         "What are the fees associated with investing in a mutual fund?",
#         "Can you recommend a mutual fund for a beginner investor?",
#         "How do I diversify my portfolio with mutual funds?",
#     ]
#     # response_contents = model_runner.run_concurrently(tasks)
#     # for response_content in response_contents:
#     #     print(response_content)
#     print(model_runner.run("How do we allocate capital efficiently in your opion Larry?"))
```