```
from swarm_models.base_llm import BaseLLM
from pydantic import BaseModel
from typing import List, Dict
import openai


class OpenRouterRequest(BaseModel):
    model: str
    messages: List[Dict[str, str]] = []


class OpenRouterChat(BaseLLM):
    """
    A class representing an OpenRouter chat model.

    Args:
        model_name (str): The name of the OpenRouter model.
                base_url (str, optional): The base URL for the OpenRouter API. Defaults to "https://openrouter.ai/api/v1/chat/completions".
        openrouter_api_key (str, optional): The API key for accessing the OpenRouter API. Defaults to None.
        system_prompt (str, optional): The system prompt for the chat model. Defaults to None.
        *args: Variable length argument list.
        **kwargs: Arbitrary keyword arguments.

    Attributes:
```

```
    model_name (str): The name of the OpenRouter model.
    base_url (str): The base URL for the OpenRouter API.
    openrouter_api_key (str): The API key for accessing the OpenRouter API.
    system_prompt (str): The system prompt for the chat model.

Methods:
    run(task, *args, **kwargs): Runs the chat model with the given task.

"""

def __init__(
    self,
    model_name: str,
    base_url: str = "https://openrouter.ai/api/v1/chat/completions",
    openrouter_api_key: str = None,
    system_prompt: str = None,
    *args,
    **kwargs,
):
    super().__init__(*args, **kwargs)
    self.model_name = model_name
    self.base_url = base_url
    self.openrouter_api_key = openrouter_api_key
    self.system_prompt = system_prompt

    openai.api_base = "https://openrouter.ai/api/v1"
```

```
    openai.api_key = openrouter_api_key


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

    Args:
        task (str): The user's task for the chat model.
        *args: Variable length argument list.
        **kwargs: Arbitrary keyword arguments.

    Returns:
        str: The response generated by the chat model.

    """
    response = openai.ChatCompletion.create(
        model=self.model_name,
        messages=[
            {"role": "system", "content": self.system_prompt},
            {"role": "user", "content": task},
        ]
        * args,
        **kwargs,
    )
    return response.choices[0].message.text
```