```
import time
import logging
from typing import Callable
from functools import wraps
from fastapi import HTTPException

# Initialize logger
logger = logging.getLogger(__name__)


def rate_limiter(max_requests: int, time_span: int, *args, **kwargs):
    """Rate limiter decorator

    Args:
        max_requests (int): Maximum number of requests allowed in the time span
        time_span (int): Time span in seconds
    """

    def decorator(func: Callable):
        last_called = {}

        @wraps(func)
        async def wrapper(*args, **kwargs):
            identifier = str(args[0].client.host)
            try:
                if identifier in last_called:
```

```
                time_diff = time.time() - last_called[identifier]
                if time_diff < time_span:
                    logger.warning(f"Rate limit exceeeded for IP: {identifier}")
                    raise HTTPException(status_code=429, detail="Too many requests")
            last_called[identifier] = time.time()
            return await func(*args, **kwargs)
        except Exception as e:
            logger.error(f"An error occurred: {str(e)}")
            raise

    return wrapper

return decorator
```