```
import concurrent.futures
import logging
import os
import warnings
from threading import Thread


def disable_langchain():
    """
    Disables the LangChain deprecation warning.
    """
    from langchain_core._api.deprecation import (
        LangChainDeprecationWarning,
    )

    # Ignore LangChainDeprecationWarning
    warnings.filterwarnings(
        "ignore", category=LangChainDeprecationWarning
    )


def disable_logging():
    """
    Disables logging for specific modules and sets up file and stream handlers.
    Runs in a separate thread to avoid blocking the main thread.
    """
```

```
os.environ["WORKSPACE_DIR"] = "agent_workspace"

warnings.filterwarnings("ignore", category=UserWarning)

# disable tensorflow warnings
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

# Set the logging level for the entire module
logging.basicConfig(level=logging.ERROR)

try:
    log = logging.getLogger("pytorch")
    log.propagate = False
    log.setLevel(logging.ERROR)
except Exception as error:
    print(f"Pytorch logging not disabled: {error}")

logger_names = [
    "tensorflow",
    "h5py",
    "numexpr",
    "git",
    "wandb.docker.auth",
    "langchain",
    "distutils",
    "urllib3",
```

```
    "elasticsearch",
    "packaging",
]

# Use concurrent futures to set the level for each logger concurrently
with concurrent.futures.ThreadPoolExecutor() as executor:
    executor.map(set_logger_level, logger_names)

# Remove all existing handlers
logging.getLogger().handlers = []

# Get the workspace directory from the environment variables
workspace_dir = os.environ["WORKSPACE_DIR"]

# Check if the workspace directory exists, if not, create it
if not os.path.exists(workspace_dir):
    os.makedirs(workspace_dir)

# Create a file handler to log errors to the file
file_handler = logging.FileHandler(
    os.path.join(workspace_dir, "error.txt")
)
file_handler.setLevel(logging.ERROR)
logging.getLogger().addHandler(file_handler)

# Create a stream handler to log errors to the terminal
```

```
    stream_handler = logging.StreamHandler()
    stream_handler.setLevel(logging.ERROR)
    logging.getLogger().addHandler(stream_handler)

    disable_langchain()



def set_logger_level(logger_name: str) -> None:
    """
    Sets the logging level for a specific logger to CRITICAL.

    Args:
        logger_name (str): The name of the logger to modify.
    """
    logger = logging.getLogger(logger_name)
    logger.setLevel(logging.CRITICAL)



def start_disable_logging_in_thread():
    """
    Starts the disable_logging function in a separate thread to avoid blocking the main thread.
    """
    thread = Thread(target=disable_logging)
    thread.start()
    return thread
```