```
# TODO: Potentially make another package for this
import json
import os
from typing import Any
import re
import shutil
import tempfile
from swarms.utils.loguru_logger import initialize_logger

logger = initialize_logger(log_folder="file_processing")


def check_if_folder_exists(folder_name: str) -> bool:
    """
    Check if a folder exists at the specified path.

    Args:
        folder_name (str): The path to the folder to check.

    Returns:
        bool: True if the folder exists, False otherwise.
    """
    try:
        return os.path.exists(folder_name) and os.path.isdir(
            folder_name
        )
```

```
    except Exception as e:
        logger.error(f"Failed to check if folder exists: {e}")
        return False


def zip_workspace(workspace_path: str, output_filename: str):
    """
    Zips the specified workspace directory and returns the path to the zipped file.
    Ensure the output_filename does not have .zip extension as it's added by make_archive.
    """
    try:
        temp_dir = tempfile.mkdtemp()
        # Remove .zip if present in output_filename to avoid duplication
        base_output_path = os.path.join(
            temp_dir, output_filename.replace(".zip", "")
        )
        zip_path = shutil.make_archive(
            base_output_path, "zip", workspace_path
        )
        return zip_path  # make_archive already appends .zip
    except Exception as e:
        logger.error(f"Failed to zip workspace: {e}")
        return None


def sanitize_file_path(file_path: str):
```

```
    """
    Cleans and sanitizes the file path to be valid for Windows.
    """
    try:
        sanitized_path = file_path.replace("`", "").strip()
        # Replace any invalid characters here with an underscore or remove them
        sanitized_path = re.sub(r'[<>:"/\\|?*]', "_", sanitized_path)
        return sanitized_path
    except Exception as e:
        logger.error(f"Failed to sanitize file path: {e}")
        return None



def load_json(json_string: str):
    """
    Loads a JSON string and returns the corresponding Python object.

    Args:
        json_string (str): The JSON string to be loaded.

    Returns:
        object: The Python object representing the JSON data.
    """
    try:
        json_data = json.loads(json_string)
        return json_data
```

```
    except json.JSONDecodeError as e:
        logger.error(f"Failed to decode JSON: {e}")
        return None


def create_file(
    content: str,
    file_path: str,
):
    """
    Creates a file with the specified content at the specified file path.

    Args:
        content (str): The content to be written to the file.
        file_path (str): The path to the file to be created.
    """
    try:
        with open(file_path, "w") as file:
            file.write(content)
        return file_path
    except Exception as e:
        logger.error(f"Failed to create file: {e}")
        return None


def create_file_in_folder(
```

```
    folder_path: str, file_name: str, content: Any
):
    """
    Creates a file in the specified folder with the given file name and content.

    Args:
        folder_path (str): The path of the folder where the file will be created.
        file_name (str): The name of the file to be created.
        content (str): The content to be written to the file.

    Returns:
        str: The path of the created file.
    """
    try:
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)

        # Create the file in the folder
        file_path = os.path.join(folder_path, file_name)
        with open(file_path, "w") as file:
            file.write(content)

        return file_path
    except Exception as e:
        logger.error(f"Failed to create file in folder: {e}")
        return None
```

```
def zip_folders(
    folder1_path: str, folder2_path: str, zip_file_path: str
):
    """
    Zip two folders into a single zip file.

    Args:
        folder1_path (str): Path to the first folder.
        folder2_path (str): Path to the second folder.
        zip_file_path (str): Path to the output zip file.

    Returns:
        None
    """
    try:
        # Create a temporary directory
        with tempfile.TemporaryDirectory() as temp_dir:
            # Copy both folders into the temporary directory
            shutil.copytree(
                folder1_path,
                os.path.join(
                    temp_dir, os.path.basename(folder1_path)
                ),
            )
```

```
        shutil.copytree(
            folder2_path,
            os.path.join(
                temp_dir, os.path.basename(folder2_path)
            ),
        )
        # Create a zip file that contains the temporary directory
        shutil.make_archive(zip_file_path, "zip", temp_dir)
except Exception as e:
    logger.error(f"Failed to zip folders: {e}")
    return None
```