Documentation Standards

1. All modules and functions must use standardized docstring format including:
    - Args
    - Returns
    - Description
    - References

Experiment Tracking

1. All experiment execution programs must:
    - Save metadata to experiments.log
    - Use standardized naming conventions for output images
    - Include experiment ID and function name in all generated image filenames

Reproducibility

1. Every training script must:
    - Call torch.manual_seed
    - Document all package versions in requirements.txt

Dependency Management

1. Newly added packages must be immediately updated in environment configuration files:
    - requirements.txt
    - Dockerfile

Testing Procedures

1. Store all test-related code and results in the tests folder
2. Test code docstrings must fully document:
    - Test purpose
    - Test date
3. Tests must run automatically

GitHub Workflow（https://github.com/JNRLEE/SBP_analyzer.git）

1. When interacting with GitHub:
    - Complete all tests before creating Pull Requests
    - Always create Pull Requests to branches first

Project Structure

The project structure should be maintained according to:/Users/jnrle/Library/CloudStorage/GoogleDrive-jenner.lee.com@gmail.com/My Drive/MicforDysphagia/ProjectDeveloper/SBP_analyzer/README.md

Import Conventions

1. Project uses a flat structure:
    - All modules (analyzer, utils, metrics, etc.) are located directly under project root directory `SBP_analyzer`.
    - ALWAYS use these import patterns:
        - `from analyzer import ...` NOT `from sbp_analyzer.analyzer import ...`
        - `import utils.file_utils` NOT `import sbp_analyzer.utils.file_utils`
    - Run pytest from the project root directory to ensure import paths work correctly.
2. All directories intended as Python packages must include an `__init__.py` file.
3. Avoid creating nested `sbp_analyzer` directory inside project root.

Function and Class Index Standards

1. All code changes must maintain up-to-date function and class indices:

    - Update the main project index (`FUNCTION_CLASS_INDEX.md`) using `function_class_index_generator.py`
    - Update module-level indices in `__init__.py` files using `update_module_index.py`
    - Follow index maintenance instructions in `CONTRIBUTION_GUIDE.md`

2. When adding or modifying classes and functions:

    - Ensure proper docstrings that follow the standardized format
    - Run index generators after making significant changes
    - Include brief descriptions that will appear in the indices

3. Module `__init__.py` files must:

    - Contain an up-to-date index of all classes and functions in the module
    - Include proper import statements and `__all__` definitions
    - Include a reference to the main index file