# Sign-Sync

## Coding Standards Document (Demo 2)





Member	Student Number
Michael Stone	u21497682
Matthew Gravette	u23545977
Wessel Johannes van der Walt	u22790919
Jamean Groenewald	u23524121
Stefan Muller	u22498622

#### **General Conventions**

#### • Language Use

Backend: Python 3.10+ (FastAPI)

Frontend: JavaScript/TypeScript (React)

Styling: Tailwind CSS for consistent UI layout

#### • File Naming

Python files: snake\_case.py

o React components: PascalCase.js

o CSS/tailwind files: component-name.module.css

#### • Variable Naming

Python: snake\_case for variables/functions, CamelCase for classes

o React/JS: camelCase for variables, PascalCase for components

#### Commenting

- o Inline comments only where necessary
- o JSDoc-style comments for complex frontend logic

### **Linting and Formatting Tools**

To maintain consistent code quality and style across the project, we use automated linting and formatting tools. These tools are enforced via GitHub Actions using <u>Super-Linter</u>.

#### **Tools Configuration**

- JavaScript/TypeScript (React)
  - o ESLint:
    - Primary linter for JS/TS code.
    - Configured via .eslintrc.js (project-specific rules).
    - Catches syntax errors, anti-patterns, and enforces code style.

#### Python

- Flake8:
  - Default linter for Python (enforces PEP 8 style guide).
  - Checks for syntax errors, undefined names, and style violations.
- o Pylint (Optional):
  - Provides deeper static analysis (e.g., code complexity, unused variables).
- o Black (Optional):
  - Opinionated code formatter (enable if adopted).

#### **GitHub Actions Setup**

The following standards are enforced on every push:

```
VALIDATE_ALL_CODEBASE: false # Only lint changed files
VALIDATE_JAVASCRIPT_ES: true # ESLint for React
VALIDATE_PYTHON_FLAKE8: true # Flake8 for Python
VALIDATE_PYTHON_PYLINT: true # Optional
VALIDATE_PYTHON_BLACK: false # Enable if using Black
```

#### Requirements

#### • Local Setup:

Developers should install and run these tools locally before pushing:
 bash

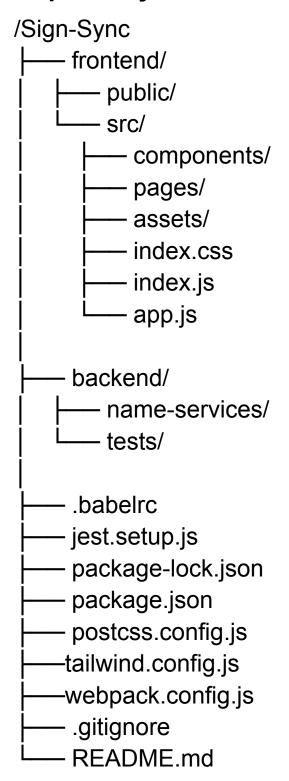
```
# JavaScript/TS

npm run lint # Runs ESLint
# Python
flake8 . # Runs Flake8
```

#### • IDE Integration:

- Configure your editor to use these tools (e.g., ESLint plugin for VSCode, Flake8 for PyCharm).
- **Pre-commit Hooks** (Recommended):
  - Add tools like pre-commit to auto-lint before commits.

## **Repository Structure**



## **Git and Branching Strategy**

Repository Type: Monorepo

This project follows a monorepo structure, where all components are maintained in a single repository.

#### **Git Branching Strategy**

This repository follows a structured branching strategy (Git-Flow):

- main The stable production-ready branch
- **develop** Integration branch for ongoing development work
- **feature/frontend/\*** Short-lived branches for individual frontend features.
- **feature/backend/\*** Short-lived branches for individual backend features.
- release/\* Temporary branches for final testing before production.
- hotfix/\* Emergency branches for critical production bug fixes.

#### **Git Organization and Management**

- Changes are introduced through feature branches and merged into main via pull requests
- Code reviews are performed before merging
- Releases follow semantic versioning (release/v1.0, release/v1.1, etc.)
- Branch naming follows a consistent pattern for easy identification (feature/, bugfix/, release/)
- Temporary test branches like Tracking\_Test are used for isolated testing
- Branches are regularly updated to stay in sync with main, as shown by the "Behind/Ahead" metrics

## **Additional Practices**

- Avoid deeply nested logic (>2 levels of indentation)
- Keep functions under 50 lines where possible
- One component per file for React
- Use .env files to store configuration secrets