### Budget Minder Project Coding Standards

To maintain code quality and ensure our code is understandable and modifiable, the following standards must be followed by all team members.

#### General Guidelines

1. **Consistent Formatting:** Use a consistent coding style across all files.
2. **Version Control:** Commit changes frequently with clear, concise commit messages.
3. **Documentation:** Document code with comments explaining complex or non-obvious sections.

#### Indentation and Spacing

1. **Indentation:** Use 2 spaces for JavaScript, HTML, and CSS; use 4 spaces for Python.
2. **Line Length:** Limit lines to a maximum of 80 characters.
3. **Spacing:** Use a single space after commas, semicolons, and around operators. No spaces inside parentheses or brackets.

#### Curly Braces

1. **Placement:** Use the "one true brace" style for braces.
2. **Blocks:** Always use curly braces for conditionals and loops.

#### Naming Conventions

1. **Variables:** Use camelCase (e.g., let userName), and use descriptive names.
2. **Constants:** Use UPPER\_SNAKE\_CASE (e.g., const MAX\_USERS).
3. **Functions/Methods:** Use camelCase (e.g., function calculateTotal()).
4. **Classes:** Use PascalCase (e.g., class BudgetManager).
5. **Files:** Use Use PascalCase

#### Commenting and Documentation

1. **Header Comments:** Each file should begin with a comment block including:
   * File name
   * Description
   * Author(s)
   * Date created
2. **Function/Method Comments:** Each should have a comment block explaining its purpose, parameters, and return value.
3. **Inline Comments:** Use sparingly for complex or non-obvious code.

#### Code Structure

1. **Modules:** Break down code into modules with clear responsibilities; use ES6 module syntax.
2. **Single Responsibility Principle:** Each function, class, or module should have a single responsibility.

#### Error Handling

1. **Exceptions:** Use try-catch blocks for exception handling.
2. **Graceful Degradation:** Ensure the application degrades gracefully in case of errors.

#### Test Coverage

1. **High Test Coverage:** Aim for high coverage, focusing on critical and complex parts of the application.

According to these standards, we aim to create a reliable and efficient code base for budget Minder. All team members must read and adhere to these guidelines when contributing to the team.