Coding Standards Document

Introduction

The objective of this document is to establish a set of coding standards, conventions, and file structure guidelines.

By adhering to these guidelines, we aim to achieve consistency, clarity, flexibility, reliability, and efficiency in our codebase. These standards play a crucial role in fostering collaboration, enhancing code readability, and facilitating code maintenance within our development team.

Coding Conventions

Naming Conventions

- When naming files:
  - For all backend files in express.js use lowercase '.' delimited file names as specified in express.js naming conventions (e.g. `class.controller.ts`, `class.repository.ts`)
  - For all other files in the fronted or .NET API use PascalCase following the (Microsoft) C# coding standard (`ClassController`, `ClassRepository`)
- For class names, use PascalCase where the first letter of each word is capitalised (e.g. `MyClass`)
- When naming functions:
  - For all functions in the front end use PascalCase which is inline with C# coding standards (e.g. `MyFunction`, `YourFunction`)
- Use camelCase for naming all variables (e.g. `myVariable`, `calculateTotal`)
- Choose descriptive and meaningful names that accurately reflect the purpose or content of the variable, function, or file.
- Avoid excessive use of abbreviations or acronyms, unless they are widely recognised and commonly used.
- Maintain consistency in naming conventions throughout the codebase

Indentation and Formatting

- Use tabs for indentation throughout the codebase.
- Utilise the K&R (Kernighan and Ritchie) style in all coding work.
    - K&R style emphasises using curly braces on their own lines, with the opening brace at the end of the line preceding the statement or control structure, and the closing brace on its own line aligned with the line of the opening brace.
    - Example:

```
if(condition)
{
  //Code block
}
```

- Add appropriate comments to clarify complex code sections or algorithms.

Comments

- Use comments to explain the purpose and functionality of the code, especially in complex or non-obvious sections.
- Write comments in clear and concise language.
- Remove or update outdated comments to avoid confusion.

File Structure

In our project, we utilise a mono repository structure to manage our codebase. This approach allows us to organise multiple related projects or modules within a single repository. The following guidelines outline the file structure conventions to be followed:

Repository Structure

- The root directory of the mono repo contains the main project configuration files and documentation.
- Each project or module is typically located in its own directory within the repository.

Project Strucure

- Inside each project or module directory, maintain a consistent structure appropriate for the project's requirements.
- Organise files and directories based on functionality, component types, or any other logical divisions.
- Consider utilising subdirectories for assets, source code, tests, configuration files, or any other relevant categories.

Linter and IDE Configurations

- Prettier Code Formatter
- ESLint