INTERNSHIP PROGRAM
2023

CODING STANDARDS

<WEB DEVELOPMENT>

<COLLEGE LISTER>

Created By:	<Ruchi Mankar>	Approved By:	<Domain Lead Name>
Created On:	29-09-2023	Approved On:	DD-MMM-YYYY

Page left blank intentionally

INDEX

General Instructions for using the Live Project Coding Standard Template

- This template and the subsequent document created using this template is a confidential document and is the intellectual property of Cloud Counselage Pvt. Ltd. Circulating it outside of the organisation without the consent of Cloud Counselage Pvt. Ltd. is the breach of company policies and will lead to legal actions
- This template is a guideline and the Developers can define their own conventions as deemed appropriate for delivering this project based on the programming language/framework they are working on.
- The **text between inequality (< >) is to be replaced** by relevant text
- Please **remove the yellow highlight on the Text** between the inequality (< >). This is done to help you notice the text to be changed/replaced
- The text in *italics* highlighted in grey is just for reference and should be removed after adding the relevant text

1. PURPOSE

The Coding Standards are the guidelines for software Developers to create uniform coding habits that eases the reading, checking and maintaining code. The intent of these standards is to define a natural style and consistency, yet leave to the authors, the freedom to practice their craft without unnecessary burden.

The coding standards shall enable the following:

- **Improve Code Quality**: Coding standards ensure that code is written consistently, readably, and maintainable manner. This makes it easier for developers to understand and work with the code, leading to higher-quality software.
- **Increase Efficiency**: By following coding standards, developers can save time by avoiding common mistakes and implementing proven solutions.
- **Facilitate Collaboration**: It creates a common language that all developers can understand and allows teams to collaborate, share code, and communicate effectively.
- **Ensure Compatibility**: It ensures that code is compatible with different platforms, browsers, and OS-device combinations.
- **Reduce Maintenance Costs**: By following established standards, developers can avoid introducing new bugs and make changes to code more quickly and easily.

The coding standards should follow the below best practices:
1. Focus on code readability
2. Enable Commenting
3. Formalising Exception Handling

2. SCOPE

This document describes general software coding standards for code written for HTML, CSS, JavaScript specific to College Lister and shall be implemented while developing the code for the said project.

3. FILE STRUCTURE

The 'File Structure' allow developers to know where files are, when to use specific code, and locate associated results. Not only do file structures streamline productivity, but they also increase code consistency and shareability

3.1. Standard File Conventions

CamelCase for HTML and JavaScript files
Uppercase names for all tables in database
Lowercase for table headers

3.2. Markdown Files

Tables in mariadb
HTML, CSS, JavaScript in Visual Studio Code
Express js was used for integration of backend and frontend

3.3. Common Conventions

Using the TAB key to create 2 spaces

4. FORMATTING CONVENTIONS

These conventions are all about the positions of line breaks, how many characters should go on a line, and everything in between.

4.1. Indentation

Using TAB for two white spaces

4.2. Using Capitalization to Aid Readability

4.3. Formatting Single Statements

4.4. Formatting Declarations

4.5. Formatting Multi-line Statements

5. NAMING CONVENTIONS

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier-for example, whether it's a constant, package, or class-which can be helpful in understanding the code.

Naming conventions result in improvements in terms of "four Cs": communication, code integration, consistency, and clarity. The idea is that "code should explain itself"

Naming convention is applicable to constants, variables, functions, modules, packages and files. In object-oriented languages, it's applicable to classes, objects, methods and instance variables.

With regard to scope, global names may have a different convention compared to local names; such as, Pascal Case for globals: Optind rather than optind in gawk. Private or protected attributes may be named differently: _secret or __secret rather than secret. Some may want to distinguish local variables from method arguments using prefixes.

For naming conventions, please refer to https://www.pluralsight.com/blog/software-development/programming-naming-conventions-explained

6. SCOPING CONVENTIONS

Scoping is generally divided into two types:

6.1. Lexical/Static Scoping

A variable in this scope always refers to its top-level environment. This characteristic of the program text has nothing to do with the call stack at runtime. Static scoping makes it considerably easier to write modular code because a programmer can find out the scope by looking at the code.

6.2. Dynamic Scoping

With dynamic scope, a global identifier directs to the identifier associated with the most current environment and is unusual in modern languages. In technical terms, each identifier has a global stack of bindings, and the most current binding is explored for events of the identifier.

In another way, the Compiler successfully explores the current block and all calling functions first in dynamic scoping.

7. COMPILE ERRORS & WARNINGS

7.1. Errors

Errors report problems that make it impossible to compile your program.

When developing programs there are three types of error that can occur:

- **Syntax error** occurs when the code given does not follow the syntax rules of the programming language. A program cannot run if it has syntax errors. Examples include:
    - misspelling a statement, e.g. writing pint instead of print
    - using a variable before it has been declared
    - missing brackets, eg opening a bracket, but not closing it

  Any such errors must be fixed first. A good integrated development environment (IDE) usually points out any syntax errors to the programmer.

- **Logic error** is an error in the way a program works. The program can run but does not do what it is expected to do. Logic errors can be caused by the programmer:

- incorrectly using logical operators, eg expecting a program to stop when the value of a variable reaches 5, but using <5 instead of <=5
- incorrectly using Boolean operators
- unintentionally creating a situation where an infinite loop may occur
- incorrectly using brackets in calculations
- unintentionally using the same variable name at different points in the program for different purposes
- using incorrect program design

- **Runtime error** is an error that takes place during the running of a program. An example is writing a program that tries to access the sixth item in an array that only contains five items. A runtime error is likely to crash the program.

7.2. Warnings

Warnings report other unusual conditions in your code that may indicate a danger points where you should check to make sure that your program really does what you intend.

Compiler warnings are useful, but they are highly unreliable. In addition, they are no substitute for language subsetting.

Please refer this article for understanding working with compiler warnings.

ENFORCING CODING STANDARD

Please refer this article to enforce coding standard across different tools and platform.