Taxi Boss

SUPREME INTERNECT

Joan Mwaniki	u16159323
Brian Ndung'u	u15322913
Dewald van Hoven	u15030378
Waldo van der Merwe	u15018556

Coding Standards Documentation

Introduction

The following document outlines the coding standards, programming styles and guidelines adhered to by Supreme Internect for the duration of the Taxi Boss project. These standards are enforced, to ensure the code is easily maintained as well as consistent.

1. Naming Conventions

a. **Naming conventions for files –** All file names will start with lowercase letters and be separated with an underscore, where file names have more than one word.

    Eg. action.php, give_rating.dart

    Not – Action.php, giveRating.dart, Give_rating.dart

b. **Naming conventions for folders–** All folder names will start with uppercase letters and be separated with an underscore, where folder names have more than one word.

    Eg. Web, Public, Wireframes

    Not – web, public, Wire_Frames

c. **Naming conventions for branches –** All branch names will start with uppercase letters and be separated with an underscore, where the branch name has more than one word.

    Eg. Public, USSD, Monitor_UI

    Not – public, ussd, monitor_ui, MonitorUI

d. **Naming conventions for functions –** All function names will use camel casing and begin with lowercase letters.

    Eg. checkEvent(), calculateAverage(), setDate()

    Not – CheckEvent(), CalculateAverage(), set_Date()

e. **Naming conventions for variables–** All variables names will use camel casing and begin with lowercase letters.

Eg. numberPlate, streetName, buttonColour

Not – Numberplate, StreetName, button_Colour

2. *Formatting Conventions*

a. ***Formatting conventions for spacing -*** *We will use Visual Studio Code's default spacing of 2.*

   i. *Spaces will be placed between variable names and operators.*
   ii. *No spaces will be placed before the semi-colon in a statement.*

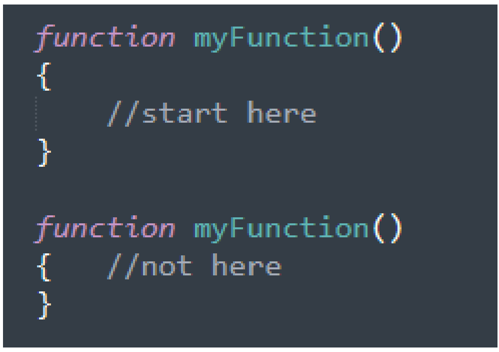
Eg. name = myName; song = happySong;

Not – name=myName; song = happySong ;

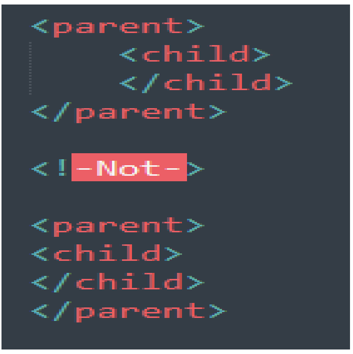
b. **Formatting conventions for indentions –** We will use Visual Studio Code's default indention of four spaces, which is equivalent to a tab.

   i. Indention will be used after braces.
   ii. Indention will be used after if statements and loops
   iii. Indentions will be used for child tags

Eg.



```
function myFunction()
{
    //start here
}

function myFunction()
{   //not here
}
```



```
<parent>
    <child>
    </child>
</parent>

<!-Not->

<parent>
<child>
</child>
</parent>
```

c. **Formatting conventions for alignment -** We will use Visual Studio Code's code alignment extension to align code. This helps improve readability.

   i. Opening and closing tags should align.
   ii. Opening and closing braces should align

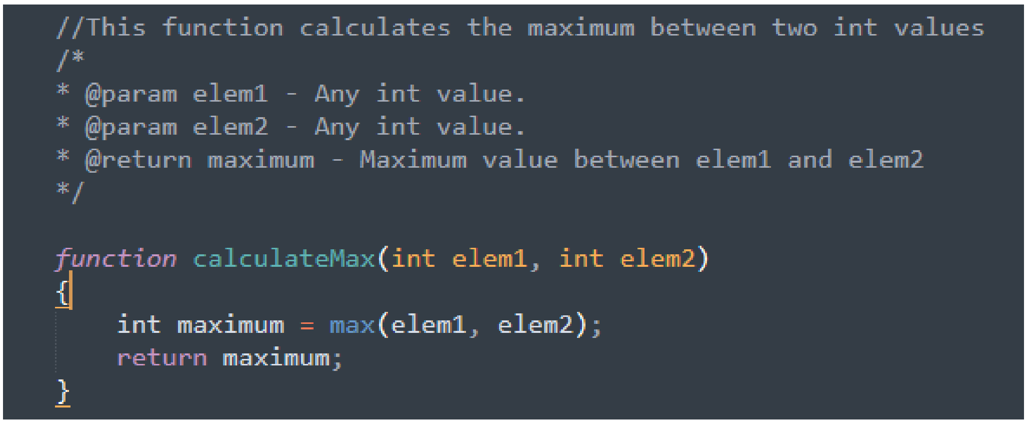
Eg. See example above.

3. Commenting conventions

a. **Inline comments –** Use comments to describe the function of a function. Add extra comments to the code when an algorithm is hard to understand.

i. Use multiline comments for long comments
ii. Use single line comments for short comments
iii. Place function description above the function implementation.
iv. For functions with parameters, use the @param tag, to describe parameters that are difficult to understand.
v. For functions with return statements, use the @return tag, to describe return values that are difficult to understand.

Eg.



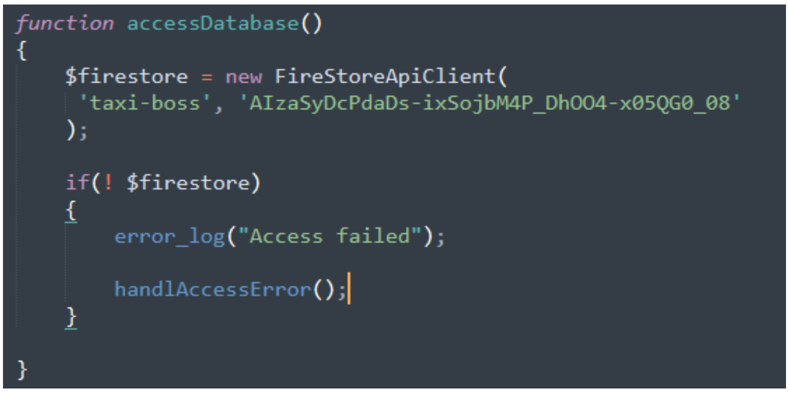
```
//This function calculates the maximum between two int values
/*
* @param elem1 - Any int value.
* @param elem2 - Any int value.
* @return maximum - Maximum value between elem1 and elem2
*/

function calculateMax(int elem1, int elem2)
{
    int maximum = max(elem1, elem2);
    return maximum;
}
```

4. Debugging and Error handling

a. Use error messages – Use error messages such as console.log() or error_log(), to display possible errors that may occur in the code.

i. Ensure error messages state the error that might have occurred
ii. Error messages should also state the line where the error occurred.
iii. Errors that have been fixed should be documented.
iv. Implement error handling methods, to handle errors.

Eg.



```
function accessDatabase()
{
    $firestore = new FireStoreApiClient(
     'taxi-boss', 'AIzaSyDcPdaDs-ixSojbM4P_DhOO4-x05QG0_08'
    );

    if(! $firestore)
    {
        error_log("Access failed");

        handlAccessError();
    }

}
```