HighTech

Coding Standards: Workflow Process and Geyser Application

By 24Fix

Janaki Patil

u16006110@tuks.co.za

Theo Naidu

u16148861@tuks.co.za

Tristan Sander-hughes

u17071390@tuks.co.za

Zi Xin Zhang

u15192556@tuks.co.za

Alexandros Mendes Petrou

u 15291792 @tuks.co.za

22 August 2019

University Of Pretoria, Hatfield
Engineering, Built environment and Information Technology

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Purpose

The purpose of these guidelines is to create uniform coding habits among the development team members. This is also so that reading verifying and maintaining code by different members is easier. The goals of these standards is to create a simple style and consistency, as well as allow the other creators free reign without any unneeded issues. General coding standards pertain to how the developer writes code. Adhering to these standards improve uniform style, clarity, flexibility, reliability and efficiency of our code

Scope

This document describes the general software standards for code that is written for the *Insurance Work Flow and Geyser App IPA* project developed by HighTech development team through the University of Pretoria for 24Fix. This includes standards for the programming languages *NodeJS, HTML, CSS, Javascript.*

Coding standards

Naming conventions

Camel casing format is used for variables, file names and function names. The ID's of the employee should be 9 characters long and should begin with 24. The agent ID should begin with A and should be 10 characters long. Collection names in the firebase should follow the camel casing format and also the function names.

Formatting conventions(use of braces)

The if and for/while loop statements and functions should have a format as below:

```
function nameOfFunction(parameter 1, parameter 2)
{
        if nameOfFunction(parameter 1, parameter 2)
        {
            // statements
        }
                    OR
        For(int I =0; i<5; i++)
        {
            //statements
        }
}
```

Brackets to be on new line and not on the same line when beginning/ending if and for loop statements as well as the functions. There should be a two line breaks after every function for readability purpose. For indentation use tab.

Commenting convention

Block comments should be placed before every function to briefly explain what the function is all and the name of the person who wrote this function.

Comments are short and straight to the point no need for long comments.

```
/*
* @author : name and email addresss
*@purpose of [ name of function]
*/
function nameOfFunction(parameter 1, prarmeter 2)
{
        if nameOfFunction(parameter 1, prarmeter 2)
        {
            // statements
        }
}
```

Variable declarations

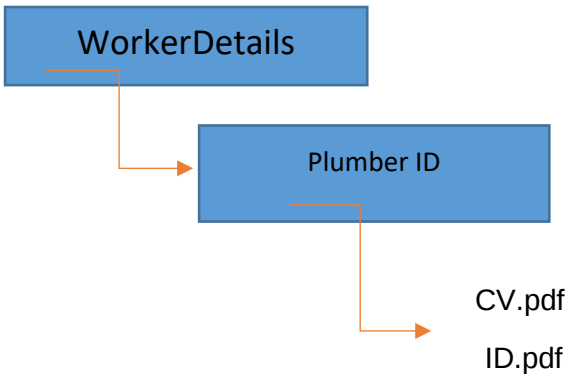
Each variable definition will be limited to one per line, and each variable will be proceeded by a type.

Rules

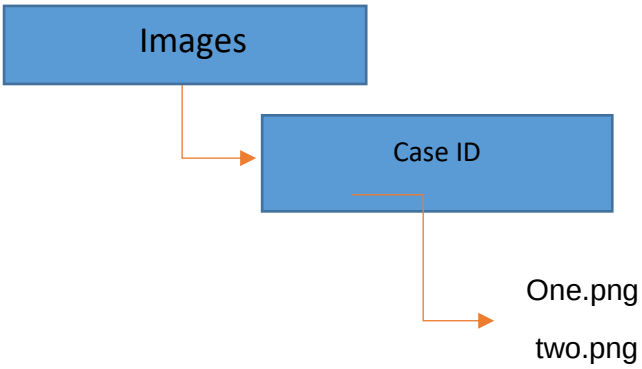
The Git structure should have the following format :

1. Master branch that contains the final version of the entire project.
2. Staging branch is used to test and make sure nothing goes wrong this is the only branch that's allowed to merge with the master branch.
3. Parent branch is used to make small changes to the program and then this branch is merged with staging.
4. Child branch are the branches where you initially start off and these branches are merged with the parent branch .

Folder to save employee documents should have the following format:



Folder to save images of installation or repairs should have the following format:



Following is the format to receive information from the client (no other format will be accepted):

Name = “ “
surname = “ “
address = “ “
cell number = “ “
call back number = “ “
client type = “ “
service type= “ “
reason = “ “

Meaningful Error Messages

Error handling is crucial in the system. Making error messages meaningful makes it easier to identify what errors have occurred due to which circumstances. These messages should also be displayed in ways that make it easy for review.
When possible, they should indicate what the problem is, where the problem occurred, and when the problem occurred. A useful Java exception handling feature is the option to show a stack trace, which shows the sequence of method calls which led up to the exception

Design patterns

Design patters are used throughout the system to improve code maintainability and modularization. This allows for addition of features without changing the entire structure of the system.

Responsibilities

1. Mr Theoveshen Naidu is assigned a role of UX designer for this system and is responsible some functionality.
2. Mr Alexandros Petrou is assigned a task to make a mobile app in ionic and manage the GitHub.
3. Mr Zi Xin Zhang is assigned a task to make an mobile app in ionic and manage the ZenHub(Project management tool).
4. Ms Janaki Patil is assigned a task to implement the core functionalities and handle documentation work.
5. Mr Tristan Sander-Hughes is assigned a task add security features to the system and perform integration testing and performing unit tests.