

Alabama Liquid Snake

University of Pretoria

Epi-Use

---

Botic - Privacy aware chatbot  
Coding Standards

---

Justin Grenfell - u16028440

Peter Msimanga - u13042352

Alicia Mulder - u14283124

Kyle Gaunt - u15330967

Lesego Mabe - u15055214

# Contents

1	Introduction	2
1.1	File Headers . . . . .	2
1.2	Class Descriptions . . . . .	2
1.3	Frontend: Angular7 . . . . .	2
1.3.1	Naming Conventions . . . . .	2
1.3.2	Formating Rules . . . . .	3
1.3.3	Commenting Practices . . . . .	4
1.3.4	Enforcement . . . . .	5
1.4	Persistence Layer . . . . .	5
1.4.1	Naming Conventions . . . . .	5
1.4.2	Formating Conventions . . . . .	5
1.4.3	Commenting Practices . . . . .	5
1.4.4	Enforcement . . . . .	5
1.5	Code Review Checklist . . . . .	5
2	References	5

# 1 Introduction

This document is meant to contain all details regarding the coding standards used in the development of all subsystems. The coding standards will be analysed on a per subsystem basis. We also define rules pertaining to the file headers as well as the description of classes.

## 1.1 File Headers

We require that our file headers have the following information:

Item	Description
File name	This is just the file name of the code file
Version number	The version number.
Author name	The name of the programmer who creates this file.
Project name	The name of the project.
Organization	The name of the team or organization that developed the project.
Related Use Cases	The shows the related use cases the resulted in the creation of this file.
Update history	List of updates, with the date of the update, the author of the changes, what was updated and why.
Reviewers	List of reviewers and what they reviewed.

Here is an example:

## 1.2 Class Descriptions

Each class will need to have a description of its purpose, a description of its methods i.e. the purpose of each, the parameters, return type, input and output, and if possible or necessary, the asymptotic complexity of the algorithm executed[1]. It should also have a description of each field, this includes the name of the field, the data type, and initialization requirement.

Here is an example:

## 1.3 Frontend: Angular7

In Angular, each component makes use of a ts file, Typescript, which is a superset of JavaScript that enforces typing, a view which uses HTML (Hypertext Markup Language) as well as styled using CSS (Cascading Style Sheets). These are going to be the focus of the coding standards defined in this subsection, as well as the directories that contain the software artifacts.

### 1.3.1 Naming Conventions

Here we have decided to makes use of the package "tslint-consistent-codestyle." This packages add many useful coding rules that are not available when using vanilla TSLint, such as the ability to control the way declare a variable, function and et cetera.[2]

The Angular CLI will be used for the creation of each component and service. New components are created using the command "ng g c" followed by the directory "components", and the subdirectory according to the function of the component, and lastly the name of the component. Here is an example:

```
1 ng generate component components/layout/footer
```

Following this rule, services and models are seperated into their respective folders, and within those folders, they are separated and ordered according their the functions.

Models, components and views are created using lowercase names. Since Angular CLI adds identifying information to the names of each file, i.e. ".component.ts" for components, ".component.html" for view,

and ".service.ts" for services, the name of the file in lower case according the name of the component, as well as it's positioning in an appropriate directory, is enough.

In the Typescript files we will require that by default, camel case is used for both method names and variables. The type is indicated using Pascal case.

### 1.3.2 Formating Rules

Here is a snippet containing some of our coding standards and additional formating rules with are checked in our code using tslint:

```
1 {
2   "rulesDirectory": [
3     "node_modules/codelyzer",
4     "tslint-consistent-codestyle"
5   ],
6   "rules": {
7     "arrow-return-shorthand": true,
8     "callable-types": true,
9     "class-name": true,
10    "comment-format": [
11      true,
12      "check-space"
13    ],
14    "curly": true,
15    "deprecation": {
16      "severity": "warn"
17    },
18    "eofline": true,
19    "forin": true,
20    "import-blacklist": [
21      true,
22      "rxjs",
23      "rxjs/Rx"
24    ],
25    "import-spacing": true,
26    "indent": [
27      true,
28      "spaces"
29    ],
30    "interface-over-type-literal": true,
31    "label-position": true,
32    "max-line-length": [
33      true,
34      140
35    ],
36    "member-access": true,
37    "member-ordering": [
38      true,
39      {
40        "order": [
41          "static-field",
42          "instance-field",
43          "static-method",
44          "instance-method"
45        ]
46      }
47    ],
48    "no-arg": true,
```

```

49     "no-bitwise": true,
50     "no-console": [
51         true,
52         "debug",
53         "info",
54         "time",
55         "timeEnd",
56         "trace"
57     ],
58     "no-construct": true,
59     "no-debugger": true,
60     "no-duplicate-super": true,
61     "no-empty": false,
62     "no-empty-interface": true,
63     "no-eval": true,
64     "no-inferrable-types": [
65         true,
66         "ignore-params"
67     ],
68     "no-misused-new": true,
69     "no-non-null-assertion": true,
70     "no-shadowed-variable": true,
71     "no-string-literal": false,
72     "no-string-throw": true,
73     "no-switch-case-fall-through": true,
74     "no-trailing-whitespace": true,
75     "no-unnecessary-initializer": true,
76     "no-unused-expression": true,
77     "no-use-before-declare": true,
78     "no-var-keyword": true,
79     "object-literal-sort-keys": false,
80 }
81 }

```

Using Visual Studio Code and having installed TSLint from the market place (<https://marketplace.visualstudio.com/items?itemName=eg2.tslint>), all the errors will be displayed whilst one is coding, however, running the command "ng lint" will display all linting errors within the project. These rules will be enforced using Prettier.

### 1.3.3 Commenting Practices

These haven't been defined just yet.

#### **1.3.4 Enforcement**

### **1.4 Persistence Layer**

#### **1.4.1 Naming Conventions**

#### **1.4.2 Formating Conventions**

#### **1.4.3 Commenting Practices**

#### **1.4.4 Enforcement**

### **1.5 Code Review Checklist**

## **2 References**

- [1] D. C. Kung, Object-Oriented Software Engineering An Agile Unified Methodology. McGraw-Hill, 2014.
- [2] A. Giretti, “Getting started with tslint and tslint-consistent-codestyle in angular 5+ projects, part 1: Visual studio code,” <https://nexusinno.com/en/getting-started-with-tslint-and-tslint-consistent-codestyle-in-angular-5-projects-part-1-visual-studio-code/>, May 2018, accessed on 2019-05-24.