



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and
Information Technology

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS301 - Software Engineering

Coviduous

Coding Standards

August 17, 2021

Team Name: CAPSlock

Team Members: * - indicates team leader

Name	Surname	Student Number
Njabulo*	Skosana*	u18089102*
Rudolf	van Graan	u16040865
Clementine	Mashile	u18139508
Peter	Okumbe	u18052640
Chaoane	Malakoane	u18202374

Contents

1	Repository Structure	4
1.1	GitHub (monorepo)	4
2	File Structure	4
2.1	Mobile Application Frontend	4
2.2	ExpressJS Backend	4
2.3	Artificial intelligence service	5
3	File Headers	5
3.1	File Name	5
3.2	Collaborators	5
3.3	File Description and Classes	5
3.4	Example	5
4	Class Description	5
4.1	Class Name	5
4.2	Class Purpose	6
4.3	Example	6
5	Enumeration Description	6
5.1	Enumeration Name	6
5.2	Enumeration Purpose	6
5.3	Example	6
6	Function Description	6
6.1	Function Name	6
6.2	Function Purpose	7
6.3	Parameters	7
6.4	Output	7
6.5	Example	7
7	Naming	7
7.1	Files	7
7.2	Class	7
7.3	Functions	7
7.4	Variables	8
7.5	Constants	8
8	Formatting	8
8.1	Indentation	8
8.1.1	Conditional Statements	8
8.1.2	Loops	8
8.1.3	Code Blocks	8

8.1.4	Block Comments	8
8.2	Spacing	9
8.2.1	Statements	9
9	Comments	9
9.0.1	In-line comments	9
9.0.2	Block comments	9

1 Repository Structure

1.1 GitHub (monorepo)

```
master
  - development
    - feature branches
```

2 File Structure

2.1 Mobile Application Frontend

```
frontend/assets
  - visual assets used by our app, such as images and fonts
frontend/lib
  - frontend
    - auth
      - authentication helper classes
    - controllers
      - classes that are used to update the models and the views
    - models
      - data structures used by the controllers
    - views
      - all screens used by the app
  globals.dart
frontend/tests
  - unit and integration tests for the frontend
```

2.2 ExpressJS Backend

```
backend/functions
  - config
    - configuration files for our database structure
  - models
    - data structures that model how data is inserted into the database
  - routes
    - HTTP routes that are called by the frontend
  - services
    - controllers that operate the backend
  - tests
    - unit and integration tests for the backend
```

2.3 Artificial intelligence service

- artificial_intelligence/covid19_prognosis_prediction_service
 - AI model
- artificial_intelligence/covid19_sourced_data
 - data used to train and test the AI

3 File Headers

Each file includes a header comment at the top to explain the name and purpose of the file, so that programmers can understand what each file is for.

3.1 File Name

The file name.

3.2 Collaborators

The file's creator and collaborators.

3.3 File Description and Classes

A description of the file, and the classes present in the file.

3.4 Example

```
3 /*  
   File name: login_screen.dart  
   Purpose: The login screen of the app. This is the entry point of Coviduous, for both admins and users.  
   Collaborators:  
     - Rudolf van Graan  
     - Clementine Mashile  
   Classes and enums:  
     - class LoginScreen extends StatefulWidget  
     - enum UserType  
     - class _LoginScreenState extends State<LoginScreen>  
3 */
```

Figure 1: File header format for Coviduous

4 Class Description

Each class within a file has a comment above it that explains its name and purpose.

4.1 Class Name

The class name.

4.2 Class Purpose

A description of the class, that explains what it does at a high level.

4.3 Example

```
/*  
  Class name: LoginScreen  
  Purpose: This class defines the route name of the login screen, so it can be  
           accessed from any other location in the app. It also defines a function for creating this screen.  
*/
```

Figure 2: Class description format for Coviduous

5 Enumeration Description

Each enumeration (enum) within a file has a comment above it that explains its name and purpose, where applicable.

5.1 Enumeration Name

The enumeration name.

5.2 Enumeration Purpose

A description of the enumeration, that explains what it does at a high level.

5.3 Example

```
/*  
  Enum name: userType  
  Purpose: Defines a user type.  
*/
```

Figure 3: Enumeration description format for Coviduous

6 Function Description

Each function within a class has a comment above it that explains its name, purpose, parameters (input), and output, where applicable.

6.1 Function Name

The function name.

6.2 Function Purpose

A description of the function, that explains what it does at a high level.

6.3 Parameters

Any parameters the function may need as input.

6.4 Output

What the function's output is.

6.5 Example

```
/*  
  Function name: _showErrorDialog  
  Purpose: Displays an error message.  
  Parameters:  
    - String msg: an error message to show  
  Output:  
    - An error message that will be displayed as a dialog box.  
*/
```

Figure 4: Function description format for Coviduous

7 Naming

7.1 Files

Files will be named in the format labelled `_like_this.dart`.

7.2 Class

Classes are named in a camel-case format, starting with an uppercase letter. For example:

```
class MyApp extends StatelessWidget {  
  doSomething();  
};
```

7.3 Functions

Functions are named in a camel-case format, starting with a lowercase letter. For example:

```
public BookOfficeSpaceResponse bookOfficeSpace() {  
  doSomething();  
}
```

7.4 Variables

Similar to functions, variables are named in a camel-case format, starting with a lowercase letter. For example:

```
public String newString;
```

7.5 Constants

Constants are named in an uppercase format, using underscores to separate words. For example:

```
private final String NEW_STRING;
```

8 Formatting

8.1 Indentation

Code will be indented using tabs to format conditional statements, loops, code blocks and block comments. Dart automatically inserts one or two tabs while typing code in IntelliJ, which we will not change. This allows the code to be more readable. For example:

8.1.1 Conditional Statements

```
if (booleanValue == true) {
    doSomething();
} else {
    doSomethingElse();
}
```

8.1.2 Loops

```
for {int i; i < 5; i++} {
    doSomething();
}
```

8.1.3 Code Blocks

```
public BookOfficeSpaceResponse bookOfficeSpace() {
    doSomething();
}
```

8.1.4 Block Comments

```
/*
    This is a block comment.
    It has more than one line.
*/
```


8.2 Spacing

In terms of spacing, there will be no space between a statement and the finishing semicolon. For example:

8.2.1 Statements

```
public String newString = "Hello";
```

9 Comments

9.0.1 In-line comments

In-line comments are written using two slashes, right above the code they comment on. For example:

```
//This is an in-line comment.  
doSomething();
```

9.0.2 Block comments

Block comments are written using a backslash followed by an asterisk to start, and an asterisk followed by a slash to end. They also appear above the code they relate to. For example:

```
/*  
    This is a block comment.  
    It has more than one line.  
*/  
doSomething();
```