Atbash Coding Standards Document

Author: Dylan Pfab, Liam Mayston

General:

Variables:

Variables should use camel case with a lowercase first letter. Dart requires private variables to begin with an underscore.

Classes:

Classes should use camel case with an uppercase first letter.

Functions:

Function names should be camel case and correspond to the functions usage.

Comments:

Comments should be used everywhere applicable. They must be precise and fully explain the code they are commenting on.

Dependencies and Imports:

Unused dependencies and imports should be removed.

Function Standards:

Use single line functions where applicable. Eg. rotate () => do something;

Use built in array functions instead of for loops. For example the map function.

Backend:

Project Structure:

```
backend/
  function/
    __tests__/
      index.test.js
```

```
    index.js
    package.json
    package-lock.json
functiontwo/
    __tests__/
        index.test.js
    index.js
    package.json
    package-lock.json
```

Formatting:

If any ambiguity/undefined behaviour is present, the WebStorm auto-formatting for Javascript should be used.

Indentation:

The ‘tab’ character is used for indentation, with 1 tab width being equivalent to 4 spaces. Code in a block is indented 1 unit deeper than its surrounding element. Code at the root level of a file has no indentation.

Spacing:

All code should be trimmed such that there is no case where 2 spaces are adjacent. In conditional and structural statements such as the ‘if’ and the ‘for’ blocks, there should be a single space between the keyword and the starting bracket. There should be a single space between the closing bracket and the curly bracket.

Use of new lines:

The length of any particular line is softly capped at 120 characters. This might not be enforced in exceptional circumstances.

Function parameters should appear on a single line along with the function name and return type. If this exceeds the 120 character per line limit, then the parameters should appear alone on separate lines.

Naming Conventions

Classes, enums and interfaces should be named using PascalCase.
Methods and variables should be named in camelCase.

Tests

The test directory should mimic the project directory. Unit and integration tests should be created for each file where testing is deemed necessary. Tests should be named to describe the test case.

Frontend:

Project Structure:

lib/
controllers/
dialogs/
    domain/
encryption/
exceptions/
    pages/
    services/
    util/
    widgets/
test/

Formatting:

Flutter formatting should be followed at all times. Use of trailing commas to encourage indentation is left up to the discretion of the developer in each case, however, readability should always be the primary concern when formatting.

Repository:

File Structure

```
.github/
    workflows/
backend/
    src/
        main/
            kotlin/
                za.ac.up.cs.atbash/
                    controller/
                    domain/
                    dto/
                    json/
                    repository/
                    security/
                    service/
            resources/
        test/
            kotlin/
                za.ac.up.cs.atbash/
                    controller/
                    service/
            resources/
mobile/
    android/
    iod/
lib/
controllers/
dialogs/
    domain/
encryption/
exceptions/
    pages/
    services/
    util/
    widgets/
test/
```

Branching Strategy

master/
  development/
    Issue-xxx
    Issue-yyy