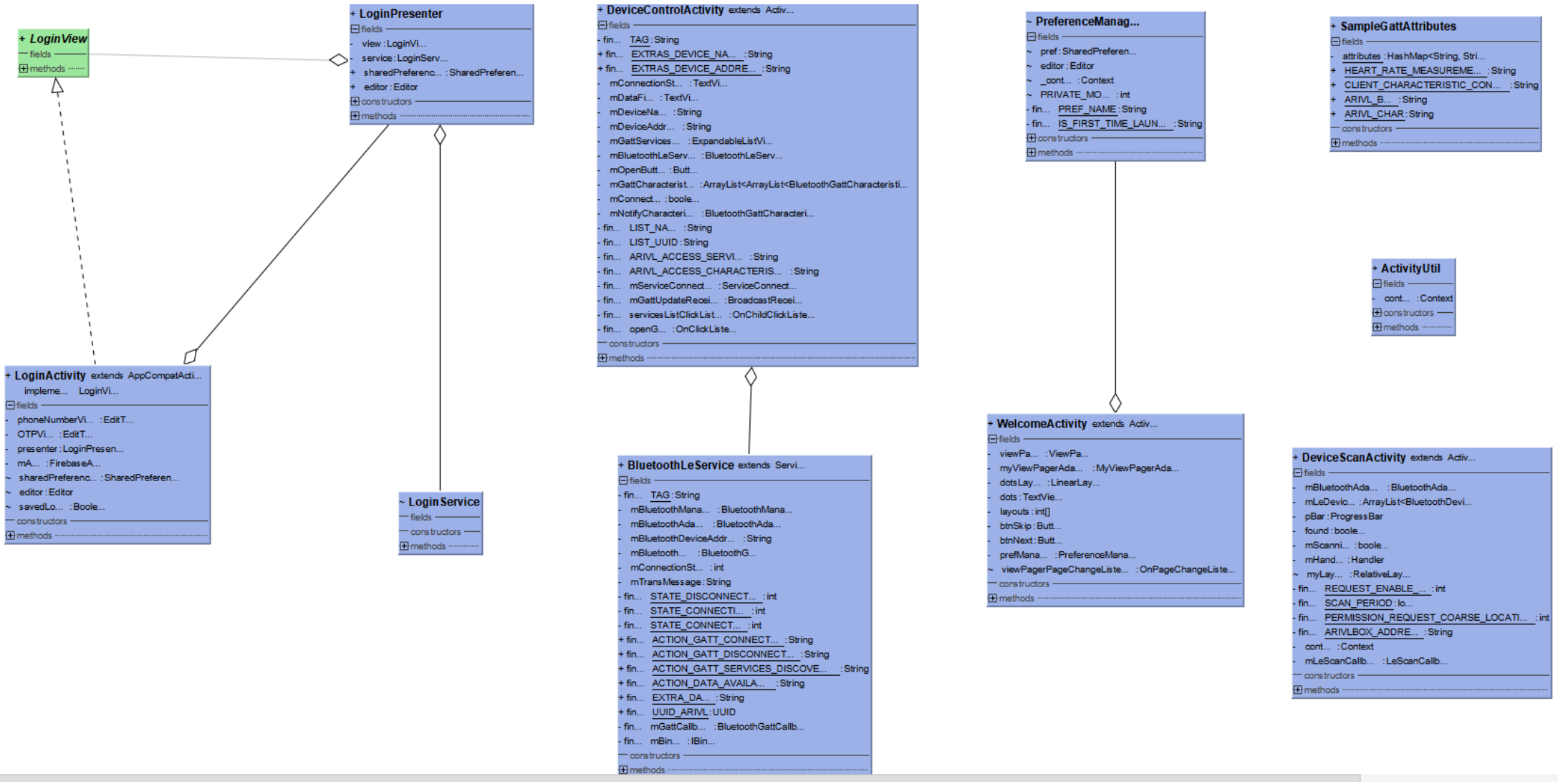
1. System design



2. Coding conventions

Variables, Methods and attributes

- Short and descriptive
- Begin with lower case and follow camel-casing convention
- Constant variables use uppercase letters

Accessors and Mutators

- Should have descriptive names
- Should have get/set prefix followed by the attribute that is being accessed

Indentation and layout

- Nested blocks should be indented
- One program statement per line
- White spaces should be used to make the code easy on the eye

Curly Brackets

- Opening curly bracket should not start on a new line
- Closing curly bracket should be on a new line
- Closing bracket and method name should have the same indentation

Exceptions

- Use relevant exceptions
- Each try block should be followed by a catch block

Imports

- Optimize imports
- Do not import using ‘import.*’

Naming conventions

1. Classes
    a. Classes follow a pascal case naming convention
    b. Class names are descriptive and short
    c. Test classes will follow a pascal case naming convention
2. Folders
    a. Names of folders will be separated using an underscore character.
    b. Names of folders will start with an uppercase letter and the first character after the underscore will also start with an underscore character
3. Variables
    a. Constant variables are all uppercase letters
    b. Variables begin with lowercase letter and the follow camel-casing convention
    c. Final variables will be uppercase characters and the words will be separated with the use of an underscore character
    d. Variables will be descriptive and the use of meaningless names will not be used

Commenting practices

- One line comments are made using double slash e.g. //
- Multiline comments are made using slash and asterisk character e.g. /*
- Multiline comments are open using a slash followed by a asterisk and the closed using an asterisk followed by a slash
- Comments will be used to provide descriptions to the reader about the classes and methods
- Every file will have a file header with the version number, author, purpose, description and the module it belongs to.
- Figure 1 is an example of how comments will be used in classes

```
/*
    Author:
    Description:
    Version number:
    Module:
*/
```

Figure 1: Example of multiline comment

File structure

- All android application files should have a ".java", ".png", and ".xml" file extension
- All Arduino files should have a ".ino" file extension
- All documentation should have a ".pdf" file extension
- Figure 2 illustrates the expanded file structure of our repository
- Figure 3 illustrates the collapsed GitHub file structure of our repository

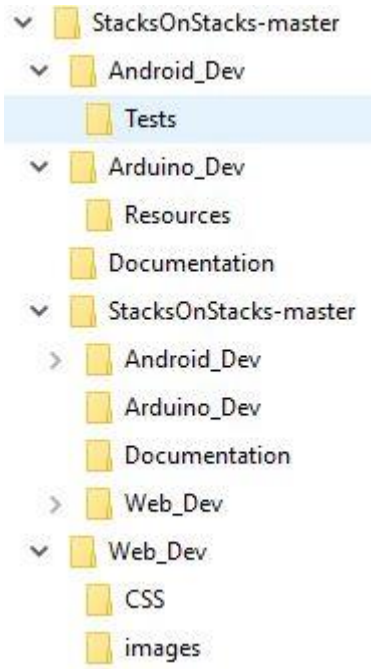


Figure 2: Expanded File Structure

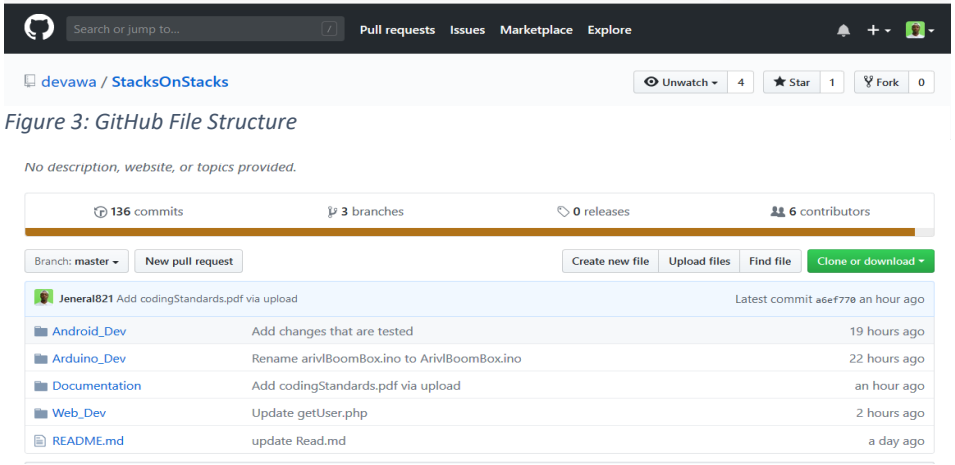


Figure 3: GitHub File Structure

Code Review process

- Android Studio provides a code scanning tool called lint that can help with identifying and correcting problems with the structural quality of the code without having to execute the app or write test cases.
- The lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.
- When using Android Studio, configured lint and IDE inspections run whenever you build your app. However, you can manually run inspections or run lint from the command line.
- The lint.xml file is a configuration file that you can use to specify any lint checks that you want to exclude and to customize problem severity levels.
- Figure 4 shows an execution of Lint in Android studios. It also shows all the errors that occur in the code

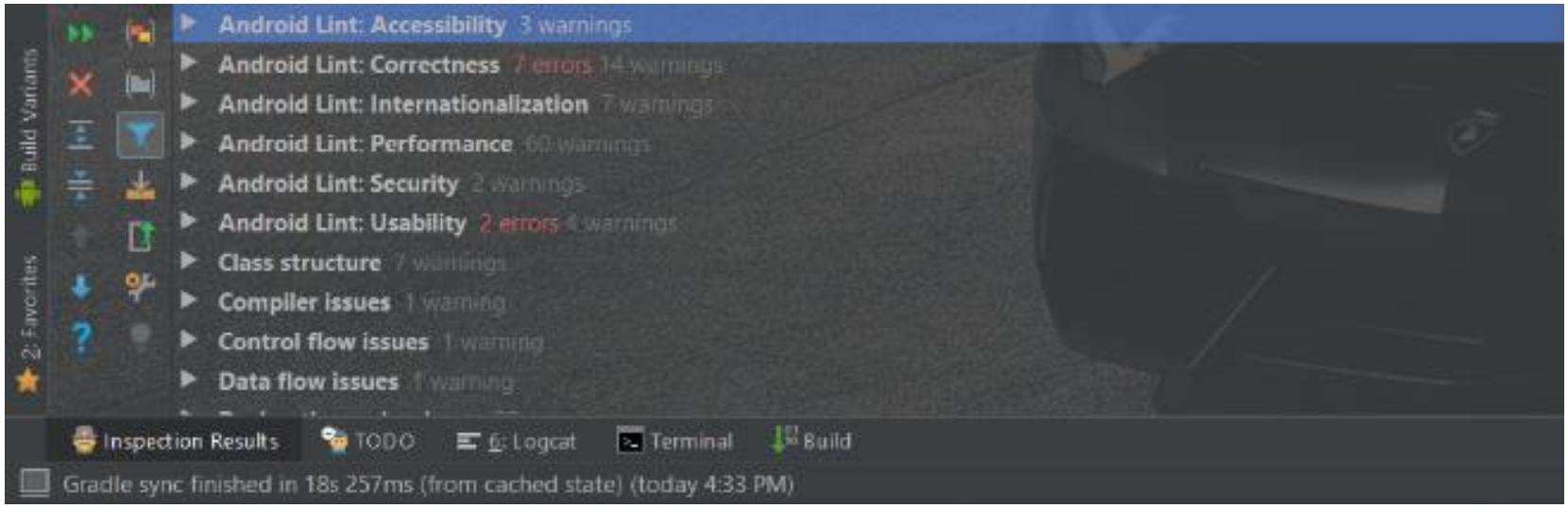


Figure 4: Lint execution