**Functionality [30 marks]**

Functionality marks will be checked by running the code, specifically with the unit tests and manual testing.

Unit tests will be shared in the workshop on testing in week 9 and we will demonstrate how to use PyTest to evaluate the functional units of your code. This is particularly applicable to the components.py functionality. Some unit tests will not be shown to students until after the feedback is returned for assessment to detect advanced features such as input validation, default value configuration in arguments, formatting of configuration files and other edge cases.

Manual testing will be executed by following the instructions you define in the README to run your code and access the user interfaces, including the command-line interfaces and the Flask web interface. We will check for typical implementation as well as handling edge cases such as how the interface responds to invalid ship placement.

**Code styling [10 marks]**

Style marks are assigned based on pylint evaluation and manually checking the source code against the criteria listed below.

* layout including consistent indentation, line length, etc.
* suitable identifier naming, appropriate variable scoping and appropriate control flow logic.
* type hinting and using default values where appropriate.
* doc strings and use of commenting.

**Project delivery [10 marks]**

Config files - Where config files are used, are they well formatted and clearly structured and contain all relevant information to customise the app for redeployment

Logging - Does the app maintain a log of events and/or errors, is the event log well structured and used throughout, is the error log stratified into classifications

Testing - Does the app test functionality through unit testing beyond what was provided?

Deployment and documentation - Is there detailed README documentation, is there a license, author and metadata listed in the documentation, is the code hosted or published with a handle in the documentation