Team 3 Code Review Guidelines

Time spent for each code review: 30 minutes.
Our code reviews are done in-person.
The writer of the code should explain their code to other team members, who will then give recommendations on how to improve the codebase.

Guidelines for Code Review:

1. Don't Repeat Yourself
    a. Do not repeat the same function call multiple times when the function call can be written once. From a maintainability perspective, if a function that has multiple parameters is called multiple times, and suddenly the function definition changes, you now have to change all of the function calls previously made.
2. Use Early Exit
    a. The more loops you have the more effort required from our mind to make sense of it. So with early exit instead of having your code wrapped in one *if loop* statement, you can just add the condition for which the loop is not executed first and just *return* if that is the case.
3. Returning Booleans
    a. If a function returns boolean, just take a closer look and see if you can write it in a better way, because most of the times, it can be written in a simpler way.
    b. Example:
        i. (BAD) If items.isEmpty is true, items.imageView = true, else, items.imageView = false.
        ii. (GOOD) items.imageView = items.isEmpty
4. Remove Unused Code
    a. Remove commented-out code that clutters the code base.
5. Write "Shy" code
    a. Don't have too many dependencies - only increase the scope of objects if you really need to.
6. Hard-coded Values
    a. Make things configurable instead of having hard-coded values. By having hard-coded values, maintaining the code becomes very difficult in the face of change.
7. Language style guide & coding conventions
    a. Your review should include consistent way of naming variables, code formatting, best practice for your language agreed on in your team.
8. Architecture/Design
    a. The code should follow the defined *architecture*. Ensure that no matter what the architecture is (MVC, MVP, MVVM or VIPER), all group members follow it. Also look out for new patterns that could be useful for the project.
    b. Check if committed code is clean code. It should follow object-oriented analysis and design (OOAD) principles or SOLID principles.
9. Unit Testing

a. Check if unit tests have been written for new features. Do they cover the failure conditions? Are they easy to read? How fragile are they?