

General Programming Standards and Guidelines Code Review Checklist

| | | | |
|----------------------------|--------------------------------|------------|---|
| Project Name | Class Attendance Mobile App | Project ID | |
| Reviewer's Name | Arielle Gabriel | Sprint No. | 4 |
| Review Date | 23 March 2018 | | |
| File Name (Source Code) | EditStudentDialogFragment.java | | |

This checklist is to be used to assess source code during a peer review. Items which represent the code being reviewed should be checked off.

Place a check mark if the code complies with the criteria. If not, place comments at the remarks column.

1. License

| Criteria | Comply | | Remarks |
|--|--------|----|---------|
| | Yes | No | |
| License exists at the top of the code | ✓ | | |
| License contains the name of the author | ✓ | | |
| License contains the following: "This is a course requirement for CS 192 Software Engineering II under the supervision of Asst. Prof. Ma. Rowena C. Solamo of the Department of Computer Science, College of Engineering, University of the Philippines, Diliman for the AY 2017-2018". | ✓ | | |

2. Code History

| Criteria | Comply | | Remarks |
|--|--------|----|---------|
| | Yes | No | |
| Code History appears after the license of the source code. | ✓ | | |
| Code History has been updated accordingly based on name of programmer, change date and change description. | ✓ | | |

3. Internal Documentation

| Criteria | Comply | | Remarks |
|---|--------|----|---|
| | Yes | No | |
| A comment block exists after the Code History containing the following information: file creation date, development group, client group and a brief statement of the purpose of the software in the file. | | ✓ | Add: each variable would have a corresponding descriptions for following vars: - args - dialogFragment - builder - inflater |
| Each method in the file is preceded by a comment block which provides the following information: method name, routine's creation date, purpose of the routine, a list of the calling arguments (their types and what they do), a list of required files and/or database tables, and return value. | ✓ | | |
| Indentions are 5 spaces. TABS ARE NOT USED. | ✓ | | |
| Inline comments are found at the top of the statement using '/*' and '*/'. | ✓ | | |
| Bracketing of block statements follows the following format: if (){ statements; } | ✓ | | |
| Exceptions are being handled appropriately. | ✓ | | |
| Meaningful variable names are used. | ✓ | | |
| Variable are initialized prior to use. | ✓ | | |
| Program statements are limited to one per line. | ✓ | | |
| Parenthesis are used to remove operator precedence ambiguity and improve code readability. | ✓ | | |
| Meaningful error messages are used. | ✓ | | |

Design Principles (Modularity, Cohesion, Coupling). Place a check mark if the method complies with Coupling and Cohesion. If not, leave it blank.

| List of Method Signature | Modularity | | Remarks |
|--|------------|----------|------------------------|
| | Coupling | Cohesion | |
| void onDialogPositiveClick(String[] studentInfo); | | | Method is an interface |
| public static EditStudentDialogFragment newInstance(String[] studentInfo); | ✓ | ✓ | |
| public void onCreate(@Nullable Bundle savedInstanceState); | ✓ | | |
| public void onAttach(Context context); | ✓ | | |
| public Dialog onCreateDialog(Bundle savedInstanceState); | | | |
| | | | |

Reviewer's Comments: