**Code Review Strategies**

- Make a checklist of which parts of the code each person will review
- Each student will review 400 lines of code which will equate to 2000 lines of code being reviewed
- The review will take 1 - 1.5hr and we will take a 15min break in the middle
- During the code review, each student will write down what needs to be fixed and constructive feedback for what the developer can improve on
- We will check for readability, maintainability, code design and functionality
- We will check for Javadoc and comments where necessary
- We will check for repeated code to keep it DRY
- We will check if the developer used good naming techniques
- Check if code follows design patterns
- Check for SOLID programming practices
- Check if we handle the exceptions and errors correctly
- Write test case, make sure it has good coverage and gives result as wanted
- Check if each method/function follows the specifications
- Check if the integration of new features works correspondingly with other methods/functions without any problems
- Check for proper author annotations in source code before the review
- Create/use a checklist as each developer goes through the code
- Verify the defects of crashes and minor to major bugs are fixed
- Not singling a developer out as it may affect the developer's mental state
- Go over official java documentation once an unfamiliar method/function was used instead of asking the developer as this may affect the all rounded test cases
- Look and check if the developer has used the optimized algorithm for his/her feature implementations
    - ** NEVER HIT : ALGORITHM ≥ O($N^2$),

      Unless Specified

- Use any testing tools as you see fits to help assist on during review
    - Point out to developer if they can and should be using a tool to help assist their development environment (e.g. using some tool to auto-generate some basic files to reduce the amount of time coding basic functionality when it could be done in less time)

Source:
https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/

**Assigned Tasks:**
Gagan: EditGrades, ProblemSetList, DialogMarks, ViewMark
Hanson: addStudent, mainActivity, mainMenu, ViewNotes
Shevlin: Editor, Browsing, CreateProblemSet, EditFileContent
Usman: PostAnnouncement, Email, StudentFileInbox, ViewAnnouncements
Julie: ViewProblems, whichProblemSet, InstructorMenu, FileView

**Summaries**

**Hanson:**
MainActivity.java

- Missing author/company signature annotation
- Use of world global variables
    - Security issues
        - Being able to access variables outside of class and project application
        - Since this is a login page, this needs to be made clear and needs to be changed immediately
- Inefficient use of imports
    - This will entitle more time to compile and waste of memory
    - Use of wildcard to import everything from the library
        - This will slow down compilation time and waste memory
- Do not initialize a variable when it is not going to be used.
    - Ex:
      id = DatabaseInsertHelper.insertNewUser(...)
      Replace above with the one below
        - DatabaseInsertHelper.insertNewUser(...)
            - This will always call and run the method regardless if you just need to initialize the database with default values.
- Variable names and method names need to be named better
    - Ex: bubblegum should be bubbleGum as there are two separate words
- Should remove some commented out lines of code
- Can see efforts were made for comments, but they need to be made and added through where necessary
- Some methods were let unused and should be removed
- No javadoc makes it harder to understand what the method is doing and accomplishing what at the end of the run time of the method

MainMenu.java

- Missing author/company signature annotation

- Inefficient use of import statements
    - Too many import statements are left unused
        - Causes the compiler allocate memory for the unused library import
- Use of world global variables
- Inconsistent with the variable name format
    - Ex nextpage should be nextPage
- Annotations for setOnClickListener(new View.OnClickListener() is missing
- Comments for specific button click listener needs to be stated

ViewNotes.java

- Missing author/company signature annotation
- Left one import statement unused
- Good use of using standardized Java syntax for variables, but should practice standard coding practices to minimize the use of global variable. This will improve the security of the application activity itself
- Coding style is marginal, left too many unused white spaces/newline characters
- No comments and Javadocs
    - Should have at least some comments in and outside the methods to show what is going on during the method call

addStudent.java

- Missing author/company
- Good use of import statements to minimize compilation phase and memory allocation
- Use world global variables
    - Security flaw
- Missing some comments and Javadocs
    - Effort was seen to add and make some comments inside the methods but was not consistent throughout

**Usman:**
PostAnnouncement,java

- No javadoc available which makes the methods not easy to understand
- Unused imports which waste of space
- No comments which makes the code hard to understand
- Global variables are not private and not instantiated which can cause security issues
- Exception e is caught which is a general exception
    - Should use a more precise exception
- Can think of a more cleaner way to loop through students

    - Instead of looping on a boolean value and waiting for loop to exit on an exception
- Check the recommended message that android studio gives for button.setOnClickListener as it suggests what to replace the new view.OnClickListener() with
- Missing author/signature of code to tell who the developer is

Email.java

- There are no comments which makes the code harder to understand
- Can use better naming techniques
    - Ex. “fab” for FloatingActionButton
- Can create a method and store the part of the FloatingActionButton and call it in the OnCreate method to clean up the look of the code
- Missing author/signature of code to tell who the developer is

StudentFileInbox.java

- There are unused import statements which is waste of space
- There are no comments which makes the code harder to understand
- No javadoc available which makes the methods not easy to understand
- Uninstantiated global variables
- Should use better naming for variables
    - Ex. myAdapter isn’t a clear name
- Code can be refactored as broken down into more methods and written in a neater fashion as it is very cluttered
- Unused method showAlert ?

ViewAnnouncements.java

- Some global variables are not private and not instantiated
- accountBalance is never used
- There are a few comments and more should be added
- No javadoc available which makes the methods not easy to understand
- Can divide work into new method and call it in the onCreate method to clean up the look

**Gaganpreet:**

EditGrades.java

- Missing comments
- Instantiate the variables (such as roleMap and displayer) globally and make them private
- Under try/catch , catch a more specific exception rather than a general one, and if there are multiple exceptions, add more catch blocks
- Display a message or print stack trace in the catch block so the debugger knows what the problem is
- Add test cases for displayMarks
- Move displayMarks function to a separate class as it should be used as a method call in the activity, so it can be changed later on without any problems
- Remove unused variables (ex. textView)  to save memory space

- Remove unused import statements
- Change the function name since the function performs a different task than what the name suggests

ProblemSetList.java

- Make variables global and private
- Add comments
- Catch a specific exception rather than a general one and if there are multiple exceptions, add more catch blocks
- Nested If statements, can be merged into one if statement to help improve readability
- Display a message or print stack trace in the catch block so the debugger knows what the problem is
- Move method to its separate class and make the activity call that method from its respective class
- The method call from DatabaseSelectHelper is being done more than once, move the call to a variable to remove repeated code
- Change method name to something that suits the task is performs

DialogMarks.java

- Add comments
- 'Context' variable is already passed on to the function as a parameter, so avoid using 'getApplicationContext()'
- Make variables private and global
- When creating the variable 'loop', the boolean value never gets changed, so just use 'True' since the while loop only exists through an exception
- Catch a specific exception rather than a general one and if there are multiple exceptions, add more catch blocks
- Nested If statements, can be merged into one if statement to help improve readability
- Instantiate the button outside of the loops and if statements, and name is according to the task it performs
- The name of the function does not match the task it performs and it matches with functions in other activities.

ViewMark.java

- Catch a specific exception rather than a general one and if there are multiple exceptions, add more catch blocks
- Add comments
- Remove unused imports
- Make variables private and global
- The method call from DatabaseSelectHelper is being done more than once, move the call to a variable to remove repeated code
- Move method to its separate class and make the activity call that method from its respective class
- Nested If statements, can be merged into one if statement to help improve readability

- When creating the variable ‘loop’, the boolean value never gets changed, so just use ‘True’ since the while loop only exists through an exception

**Shevlin:**

Overall:

- S from Solid might be possible if similar methods in the code get their own classes
- Missing Comments

Editor.java

- Inefficient use of imports
    - It will cause the compiler to allocate more memory for the unused libraries
- Reduce the number of global variables to increase security for the application
- In the ‘question’ button listener, the functionality can be splitted into 3 separate methods:
    - Write to a file
    - Read to a file
    - File upload
        - File upload can be splitted into 2 to 3 different independent methods
    - Please review standardized coding practices to split up independent functionality into their own methods
- There should not be a need to check the SDK version of android to check and add permissions to write and read from internal storage system on android
- Empty methods
    - beforeTextChanged, onTextChanged
- Possibly refactor code to merge the 2 try blocks for code organization and good coding style
- Long lines of code (could be easier to read)

Browsing.java

- Unused variables
- Commented out code (Either use or delete for readability and organization)
- Normal Exception e catch could be more precise in order to make debugging any exceptions easily
- Unused methods can be removed for code readability and organization

CreateProblemSet.java

- Unused variables, can be removed for code readability and organization
- Unused functions can be removed for code readability and organization
- Method separation at the end (Code readability, methods should be separated by at least 1 newline)
- Sys.out.prln in android code? (android prints are handled using either logs or the UI)

EditFileContent.java

- Variable declarations not centralized
- Commented out code and should be removed

- In the floatingActionButton listener, the functionality can be splitted into 3 separate methods:
    - Write to a file
    - Read to a file
    - File upload
        - File upload can be splitted into 2 to 3 different independent methods
- There should not be a need to check the SDK version of android to check and add permissions to write and read from internal storage system on android
- Code needs to be kept code
    - Too many copy and pasting of code
        - Redesigning of the code is needed
- Empty methods
    - beforeTextChanged, onTextChanged
- Long line of code (hard to read)
    - Add more comments to explain the methods and explain during the internal calls
    - Should be kept short
        - Please review standardized coding practices

**Julie**

FileView.java:

- No javadoc
- No comment

InstructorMenu.java:

- Need more javadoc
- Need more comment
- Need to remove commented out code
- Line 60 typo: “Boolean”, should be “boolean”
- getPath, String method with “return null” at end
- getDataColumn, String method with “return null” at end
- Starting line 341: 4 methods can be put into one method, checking file type
- Long lines of code

ViewProblems.java:

- No comment
- No javadoc
- Parameters should have more meaningful names: “newList”, “temp”, “temp2”
- Try-catch block: generic exception
- Long lines of code

WhichProblemSet.java:

- No javadoc
- No comment
- Check method: unused flag check