Code Review of
‘feature-rich desktop calendar software’

Course Name: **Software Development Project**

Course No: **CSE 3106**

Submitted to:
Dr. Amit Kumar Mondal
Associate Professor
Computer Science & Engineering Discipline,
Khulna University,
Khulna.

Submitted by:

Name: Muhammad Fahim
Student ID: 210210

Name: Umme Talha
Student ID: 210223

Project Title: feature-rich desktop calendar software

Project Developers:

Name: Sharmika Das Banhi
Student ID: 210204

Name: Redwan
Student ID: 210207

Code Reviewed By:

Name: Muhammad Fahim
Student ID: 210210

Name: Umme Talha
Student ID: 210223

Generic Checklist for Code Review of 'feature-rich desktop calendar software'

Generic Checklist for Code Reviews:

Structure

	Description of Item	Yes	No	
1	Does the code completely and correctly implement the design?		✓	
2	Does the code conform to any pertinent coding standards?	✓		
3	Is the code well-structured, consistent in style, and consistently formatted?		✓	
4	Are there any uncalled or unneeded procedures or any unreachable code?	✓		
5	Are there any leftover stubs or test routines in the code?	✓		
6	Can any code be replaced by calls to external reusable components or library functions?	✓		
7	Are there any blocks of repeated code that could be condensed into a single procedure?		✓	
8	Is storage use efficient?	✓		
9	Are symbolics used rather than "magic number" constants or string constants?		✓	
10	Are any modules excessively complex and should be restructured or split into multiple routines?		✓	

Documentation

	Description of Item	Yes	No	
1	Is the code clearly and adequately documented with an easy-to-maintain commenting style?		✓	
2	Are all comments consistent with the code?	✓		

Variables

	Description of Item	Yes	No	
1	Are all variables properly defined with meaningful, consistent, and clear names?	✓		
2	Do all assigned variables have proper type consistency or casting?	✓		
3	Are there any redundant or unused variables?		✓	

Structure

	Description of Item	Yes	No	
1	Does the code follow the style guide for this project?	✓		
2	Is the header information for each file and each function descriptive enough?		✓	
3	Is there an appropriate number of comments? (frequency, location, and level of detail)		✓	

4	Is the code well structured? (typographically and functionally)	✓		
5	Are the variable and function names descriptive and consistent in style?	✓		
6	Are "magic numbers" avoided? (use named constants rather than numbers)		✓	
7	Is there any “dead code” (commented out code or unreachable code) that should be removed?	✓		
8	Is it possible to remove any of the assembly language code, if present?		✓	
9	Is the code too tricky? (Did you have to think hard to understand what it does?)	✓		
10	Did you have to ask the author what the code does? (code should be self-explanatory)	✓		

Architecture

	Description of Item	Yes	No	
1	Is the function too long? (e.g., longer than fits on one printed page)	✓		
2	Can this code be reused? Should it be reusing something else?		✓	
3	Is there minimal use of global variables? Do all variables have minimum scope?	✓		
4	Are classes and functions that are doing related things grouped appropriately? (cohesion)	✓		
5	Is the code portable? (especially variable sizes, e.g., “int32” instead of “long”)		✓	
6	Are specific types used when possible? (e.g., “unsigned” and typedef, not just "int")		✓	

7	Are there any if/else structures nested more than two deep? (consecutive “else if” is OK)	✓		
8	Are there nested switch or case statements? (they should never be nested)		✓	

Variables

	Description of Item	Yes	No	
1	Does the code avoid comparing floating-point numbers for equality?	N/A	N/A	
2	Does the code systematically prevent rounding errors?	N/A	N/A	
3	Does the code avoid additions and subtractions on numbers with greatly different magnitudes?	N/A	N/A	
4	Are divisors tested for zero or noise?	N/A	N/A	

Loops and Branches

	Description of Item	Yes	No	
1	Are all loops, branches, and logic constructs complete, correct, and properly nested?	✓		
2	Are the most common cases tested first in IF- -ELSEIF chains?			

		N/A	N/A	
3	Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?	N/A	N/A	
4	Does every case statement have a default?	N/A	N/A	
5	Are loop termination conditions obvious and invariably achievable?	✓		
6	Are indexes or subscripts properly initialized, just prior to the loop?	N/A	N/A	
7	Can any statements that are enclosed within loops be placed outside the loops?	N/A	N/A	
8	Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?	N/A	N/A	

Maintainability

	Description of Item	Yes	No	
1	Does the code make sense?	✓		
2	Does the code comply with the accepted Coding Conventions?	✓		
3	Does the code comply with the accepted Best Practices?		✓	
4	Does the code comply with the accepted Comment Conventions?		✓	

5	Is the commenting clear and adequate?		✓	
6	Are ideas presented clearly in the code?	✓		
7	Is encapsulation done properly?	✓		
8	Is the code not too complex?		✓	
9	Are there no unnecessary global variables?		✓	
10	Is the reading order in source code from top to bottom?	✓		
11	Are there unused variables or functions?		✓	

Reusability

	Description of Item	Yes	No	
1	Are all available libraries being used effectively?	✓		
2	Are available **OpenMRS** util methods known and used?	N/A	N/A	
3	Is the code as generalized/abstracted as it could be?		✓	
4	Is the code a candidate for reusability?		✓	

Robustness

	Description of Item	Yes	No	
1	Are all parameters checked?	✓		
2	Are error conditions caught?	✓		
3	Is there a default case in all switch statements?	✓		
4	Is there a non-reentrant code in dangerous places?	✓		
5	Is the usage of macros proper? (Readability, complexity, portability...)	✓		
6	Is there unnecessary optimization that may hinder maintainability?		✓	

Error Handling

	Description of Item	Yes	No	
1	Does the code comply with the accepted Exception Handling Conventions?		✓	
2	Does the code make use of exception handling?	✓		

3	Does the code simply catch exceptions and log them?	N/A	N/A	
4	Does the code catch general exception?	N/A	N/A	
5	Does the code correctly impose conditions for "expected" values?	N/A	N/A	
6	Are input parameters checked for proper values (sanity checking)?	N/A	N/A	
7	Are error return codes/exception generated and passed back to the calling function?	N/A	N/A	
8	Are error return codes/exceptions handled by the calling function?	N/A	N/A	
9	Are null pointers and negative numbers handled properly?	N/A	N/A	
10	Do switch statements have a default clause used for error detection?	N/A	N/A	
11	Are arrays checked for out-of-range indexing? Are pointers similarly checked?	N/A	N/A	
12	Is garbage collection being done properly, especially for errors/exceptions?	N/A	N/A	
13	Is there a chance of mathematical overflow/underflow?	N/A	N/A	

14	Are error conditions checked and logged? Are the error messages/codes meaningful?	N/A	N/A	
15	Would an error handling structure such as try/catch be useful? (depends upon language)	N/A	N/A	