

# **Code Review of The Software Project: Money Manager application**

Course Title: Software Development Project  
Course ID: CSE- 3106

**Project By:**

SK Miraz Rahman Ani  
Student ID: 210211  
Mohammad IbnSina  
Student ID: 200210

**Reviewed By:**

M.d. Ashiquzzaman Rahad  
Student ID: 210201  
Jannatul Ferdous Nijhum  
Student ID: 210239

**Submitted To:**

Amit Kumar Mondol  
Associate Professor  
Computer Science & Engineering Discipline  
Khulna University,  
Khulna.

# Introduction

This code review evaluates the Money Manager application, including the login, registration, and dashboard, summary, income and expense sections. The review identifies areas for improvement, adherence to best practices, and suggestions for enhancing maintainability, security, and overall code quality. In this code review, bad smells of the code, architecture evaluation, modularity check, condition statements of the code & other related sections are evaluated.

## Code Smells

### 1. Large or complex methods:

There are not many large or complex methods which can be difficult to read and understand. The average method size is close to 10 lines of code which is close to the standard size of methods. The highest size of method is in expense.py module with around 24 lines of code (submit\_button\_press method).

### 2. Long parameter lists:

There is no method with long parameter lists.

### 3. Excessive comments:

In some modules there are excessive comments and some doesnot have any comment at all. In the dashboard.py, there are enough comments. But in the expense.py, income.py, registration.py and summary.py modules, there is no comment at all. So, there are inconsistencies in using comments.

### 4. Duplicate code:

There are some duplicate code in the similar kind of methods. But other than that code reusability has been done quite effectively in most of the modules.

### **5. Inconsistent naming conventions:**

In most cases, naming conventions are standardized. For example: `switch_income()`, `switch_login()`, `relative_to_assets()`, etc.

### **6. Incomplete error handling:**

Errors are mostly handled in each of the modules. There are mainly scope of running into error while doing file operations which seems to be handled carefully.

### **7. Too many if/else statements:**

In the `dashboard.py`, `expense.py` and `income.py` modules, there are some excessive use of if/else statements. But in other modules, the usage of if/else statements are moderated.

### **8. Poor use of inheritance:**

There is no poor use of inheritance in this project which is particularly causing any problem.

### **9. Unnecessary dependencies:**

No third party libraries or frameworks are used here. So there is no unnecessary dependencies on the code.

### **10. Magic numbers or hard-coded values:**

Some magic numbers are used in the code for different functional works. For example, in `dashboard.py`, magic numbers are used for drawing pie chart purpose.

## **Proposed Architecture Evaluation**

The proposed architecture of the project is “**Repository Architecture**”. In the project, we can see the reflection of the proposed architecture. There are different layers or modules in the project.

**1. Centralized Data Management:** A repository architecture allows for centralizing all financial data related to accounts, transactions, budgets, and other relevant information. This centralized approach simplifies data management, ensuring consistency and reducing the likelihood of data discrepancies.

**2. Abstraction of Data Access:** By employing a repository pattern, the data access logic is abstracted away from the rest of the application. This separation of concerns makes the codebase more maintainable and flexible, as changes to the underlying data storage technology can be made without impacting the business logic.

**3. Scalability and Flexibility:** A well-designed repository architecture facilitates scalability by providing a modular structure that can accommodate changes in data volume and system complexity. As the money management system grows, additional repositories can be added or existing ones modified to support new features and requirements.

**4. Interoperability and Integration:** Repositories provide a standardized interface for accessing and manipulating data, making it easier to integrate the money management system.

So, we can say that the project reflects the proposed **Repository Architecture** more or less.

## Modularity Check

The project is divided into 6 modules which are **dashboard.py**, **income.py**, **expense.py**, **registration.py**, **login.py** & **summery.py**.

**login.py** module has the main window and frames of the user interface where user can login with their email and password. If the user is not registered yet, the system moves the user to **registration.py** module which serves user to register with their name, email and password. In

**dashboard.py** module, the user can see balance, the overview of the system of their income and expense details in pie chart. **income.py** module serves the user to input income source and income amount and **expense.py** serves to take input of expense source and expense amount from the user. In **summery.py** modules, the user can see the summery of his income and expense details according to date.

## **If/else Condition to Switch statement**

There is no built-in switch case statement in python. As a result, there is no other way to implement the conditions required in the code other than using if/else statement.



The Code Review Checklist provides a company guideline for checking code including pass/fail parameters and recording any comments when the test fails.

During a project, this document is used by team members as follows:

- 1 During project planning, it is utilized as a reminder for how much review time should be allocated during the project for the software being developed
- 2 During design and development (coding) portion of the project, the checklists are used to conduct code reviews.

*The list of test items is representative and should be modified prior to use to reflect your development environment and standards.*

### Generic Checklist for Code Reviews

#### Structure

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code completely and correctly implement the design?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code conform to any pertinent coding standards?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the code well-structured, consistent in style, and consistently formatted?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any uncalled or unneeded procedures or any unreachable code?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are there any leftover stubs or test routines in the code?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Can any code be replaced by calls to external reusable components or library functions?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any blocks of repeated code that could be condensed into a single procedure?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is storage use efficient?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are symbolics used rather than "magic number" constants or string constants?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are any modules excessively complex and should be restructured or split into multiple routines?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

#### Documentation

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Is the code clearly and adequately documented	<input checked="" type="checkbox"/>	<input type="checkbox"/>	



<input type="checkbox"/>	with an easy-to-maintain commenting style?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are all comments consistent with the code?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

### Variables

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Are all variables properly defined with meaningful, consistent, and clear names?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Do all assigned variables have proper type consistency or casting?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any redundant or unused variables?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

### Style

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code follow the style guide for this project?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the header information for each file and each function descriptive enough?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is there an appropriate amount of comments? (frequency, location, and level of detail)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is the code well structured? (typographically and functionally)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are the variable and function names descriptive and consistent in style?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are "magic numbers" avoided? (use named constants rather than numbers)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is there any "dead code" (commented out code or unreachable code) that should be removed?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is it possible to remove any of the assembly language code, if present?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is the code too tricky? (Did you have to think hard to understand what it does?)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Did you have to ask the author what the code does? (code should be self-explanatory)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

### Architecture

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Is the function too long? (e.g., longer than fits on one printed page)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Can this code be reused? Should it be reusing something else?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	



<input type="checkbox"/>	Is there minimal use of global variables? Do all variables have minimum scope?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are classes and functions that are doing related things grouped appropriately? (cohesion)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is the code portable? (especially variable sizes, e.g., "int32" instead of "long")	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are specific types used when possible? (e.g., "unsigned" and typedef, not just "int")	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any if/else structures nested more than two deep? (consecutive "else if" is OK)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are there nested switch or case statements? (they should never be nested)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

### Arithmetic Operations

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code avoid comparing floating-point numbers for equality?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code systematically prevent rounding errors?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code avoid additions and subtractions on numbers with greatly different magnitudes?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are divisors tested for zero or noise?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

### Loops and Branches

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Are all loops, branches, and logic constructs complete, correct, and properly nested?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are the most common cases tested first in IF- -ELSEIF chains?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does every case statement have a default?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are loop termination conditions obvious and invariably achievable?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are indexes or subscripts properly initialized, just prior to the loop?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Can any statements that are enclosed within loops be placed outside the loops?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	





Defensive Programming				
	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Are indexes, pointers, and subscripts tested against array, record, or file bounds?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are imported data and input arguments tested for validity and completeness?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are all output variables assigned?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are the correct data operated on in each statement?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is every memory allocation deallocated?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are timeouts or error traps used for external device accesses?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are files checked for existence before attempting to access them?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are all files and devices left in the correct state upon program termination?	<input type="checkbox"/>	<input type="checkbox"/>	

Maintainability				
	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code make sense?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code comply with the accepted Coding Conventions?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code comply with the accepted Best Practices?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code comply with the accepted Comment Conventions?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the commenting clear and adequate? (As a guide, each file will have a comment at the start, explaining what the code does, possibly a comment at the start of each function, and comments as needed to explain complex or obfuscated code.)			
<input type="checkbox"/>	Are ideas presented clearly in the code?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is encapsulation done properly?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Is the code not too complex?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there no unnecessary global variables?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the reading order in source code from top to bottom?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there unused variables or functions?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	



### Requirements and Functionality

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code match the requirements, specifications and standards?	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	Is the logic proper? Does the code function as needed?	<input type="checkbox"/>	<input type="checkbox"/>	

### System and Library Calls

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Do all system calls have their return status checked?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are all possible errors from system or library calls handled?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are signals caught and handled?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is mutex() used on multithreaded code on global variables?	<input type="checkbox"/>	<input type="checkbox"/>	

### Reusability

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Are all available libraries being used effectively?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are available openmrs util methods known and used?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the code as generalized/abstracted as it could be?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the code a candidate for reusability?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

### Robustness

	Description of Item	Pass	Fail	Comments
	Are all parameters checked?	<input checked="" type="checkbox"/>		
	Are error conditions caught?		<input checked="" type="checkbox"/>	
	Is there a default case in all switch statements?	<input checked="" type="checkbox"/>		
	Is there non-reentrant code in dangerous places?		<input checked="" type="checkbox"/>	
	Is the usage of macros proper? (Readability, complexity, portability...)	<input checked="" type="checkbox"/>		
	Is there unnecessary optimization that may hinder maintainability?		<input checked="" type="checkbox"/>	



Security				
	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code appear to pose a security concern?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Do Service methods have an @Authorize annotation on them	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the application use an inclusion list (known, valid, and safe input) rather than an inclusion list (rejecting known malicious or dangerous input)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is all user input encoding set by the server?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is all character encoding set by the server?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	If cookies contain sensitive data, are they marked secure?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Do input surfaces in Web parts and other customizations include boundary checks, input data integrity checks, and appropriate exception handling to protect from cross-site scripting and SQL injection.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the design address potential canonicalization issues?	<input type="checkbox"/>	<input type="checkbox"/>	

Control Structures				
	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the application log sensitive data in clear text.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Sensitive data is not stored in cookies.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Sensitive data is not stored in unencrypted, hidden form fields or query strings. It is maintained by using server-side state management.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	SSL, IPSEC with encryption, or application layer encryption prior to transmittal is used to protect sensitive data during transmission.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Sensitive data is not cached. Output caching is off by default.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Sensitive data that is transferred via email uses S/MIME encryption or Information Rights Management (IRM), depending upon the intended recipient.	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code make use of infinite loops?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the loop iterate the correct number of times?	<input type="checkbox"/>	<input type="checkbox"/>	



### Resource Leaks

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code release resources?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code release resources more than once?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code use the most efficient class when dealing with certain resources?	<input type="checkbox"/>	<input type="checkbox"/>	

### Error Handling

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Does the code comply with the accepted Exception Handling Conventions?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code make use of exception handling?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code simply catch exceptions and log them?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code catch general exception (java.lang.Exception)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Does the code correctly impose conditions for "expected" values?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are input parameters checked for proper values (sanity checking)?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are error return codes/exception generated and passed back to the calling function?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are error return codes/exceptions handled by the calling function?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are null pointers and negative numbers handled properly?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Do switch statements have a default clause used for error detection?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are arrays checked for out of range indexing? Are pointers similarly checked?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is garbage collection being done properly, especially for errors/exceptions?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is there a chance of mathematical overflow/underflow?	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Are error conditions checked and logged? Are the error messages/codes meaningful?	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Would an error handling structure such as try/catch be useful? (depends upon language)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	



### Timing

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Is the worst case timing bounded? (no unbounded loops, no recursion)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any race conditions? (especially multi-byte variables modified by an interrupt)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is appropriate code thread safe and reentrant?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are there any long running ISRs? Are interrupts masked for more than a few clocks?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is priority inversion avoided or handled by the RTOS?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the watchdog timer turned on? Is the watchdog kicked only if every task is executing?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Has code readability been sacrificed for unnecessary optimization?	<input type="checkbox"/>	<input type="checkbox"/>	

### Validation & Test

	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Is the code easy to test? (How many paths are there through the code?)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Do unit tests have 100% branch coverage? (code should be written to make this easy)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are the compilation and/or lint checks 100% warning-free? (Are warnings enabled?)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is special attention given to corner cases, boundaries, and negative test cases?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the code provide convenient ways to inject faulty conditions for testing?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are all interfaces tested, including all exceptions?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Has the worst case resource use been validated? (stack space, memory allocation)	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are run-time assertions being used? Are assertion violations logged?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is there commented out code (for testing) that should be removed?	<input type="checkbox"/>	<input type="checkbox"/>	



Hardware				
	Description of Item	Pass	Fail	Comments
<input type="checkbox"/>	Do I/O operations put the hardware in a correct state?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Are min/max timing requirements met for the hardware interface?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is it ensured that multi-byte hardware registers can't change during read/write?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Does the software ensure that the system resets to a well defined hardware system state?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Have brownout and power loss been handled?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Is the system correctly configured for entering/leaving sleep mode (e.g. timers)?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Have unused interrupt vectors been directed to an error handler?	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	Has care been taken to avoid EEPROM corruption? (e.g., power loss during write)	<input type="checkbox"/>	<input type="checkbox"/>	