# CO 3256 - QUALITY ENGINEERING

# ASSIGNMENT-02

NAME - U.D.P.D UDUWELA

INDEX NO. - 18/ENG/112

REGISTRATION NO. - EN91361

# CONTENTS

## I. INTRODUCTION

Many aspects of the world have altered and improved due to technological advancements. When comparing with Sri Lanka, though Sri Lanka has developed with some fields, the bill reading system of electricity still has not changed. The system follows the manual system yet. So, I decided to change the bill reading system from manual to automatically. So, the main purpose of this system is to read the current bill by the user himself without any complex calculations. From this system the usage of current system can be reduced. As well as, from this self-care app both user and the system provider are updated with the current usage and the overall usage. From this system, it is decided to perform both bill reading and bill paying system from only one system. So, this software can be known as two in one software. The system consists of both hardware parts and software parts. The software part consists of a responsive web application for the Customer and service provider. But in this project, I have developed web applications and its services only for the customer. From the hardware part, the used units of the device are calculated in a cloud server. The result from the hardware part is taken from the software part, and from the software, the output or the usage units and the bill is shown to the user. So, both the hardware part and the software part have interacted with each other well. Therefore before using the software, the hardware part should be fixed into our main trip switchboard. Then there should be a proper wifi connection to connect the smart meter device to the cloud database.

Though the bill paying and bill reading system have not been updated yet, people have to face some problems. The problems are shown below.

- With the current system, lots of manpower is wasted for a manual meter reading.
- Customers are not able to view their current usage easily.
- If the usage passes a certain limit, additional charges are applied to the bill.
- People are not aware of peak and off-peak times of electricity usage.
- There's no single platform for customers to manage and pay their electricity bill using mobile applications.

Since these are very common issues among all the customers in Sri Lanka as well as all around the world this project can have an impact on all the people who own a smartphone and a payment card, which is quite a larger percentage. The smart meter system has three main sub systems. They are,

- Registering, login, and authentication system
- The card payment verification system
- Security system Above three subsystems are interconnected together to create our full system.

In the registering, the customer's account numbers regarding the CEB manual bill are taken first. After verifying the account, the details of the customer are taken. After the verification succeed, the account is created successfully. After creating the account unique user name and a password is provided to the customer. In the future login, the customer can log with the account by entering the user name and password. In the account, the account details of the electricity bill are shown separately. They are electricity side, the current usage, overall usage and the calculated bill is shown in the account properly. On the other hand, through this system, user can control his / her home electric appliances anywhere at any time. If the device is not working or has any issue, it will also show the CEB live meter map and also popup error messages if the meter is in inactive status. The location longitude and latitude values are observed from the neo-6m GPS module.If the user wants to complain about anything user can use the email sending function through the web application. In this system al the unit calcultion and total bill calculation part is done within the server .

## II. MCCALL'S QUALITY FACTORS THAT INVOLVED IN TESTING THE QUALITY OF THE PROJECT

Quality factors represent a system's behavioral feature. They are characteristics of a software system that are external to it. In McCall's quality factors, there are eleven quality factors. Correctness, dependability, efficiency, integrity, usability, maintainability, testability, adaptability, portability, reusability, and interoperability are the characteristics they possess. The quality elements are chosen for the CEB Self Care APP, and an explanation of each is listed below.

### ➢ Correctness

Correctness means how much a program satisfies its specifications and fulfills the user's mission objectives. In CEB Self-care App, correctness refers to giving correct electricity usage statistics in real-time and generating the bill for units that are consumed at any time for each separate user. Even if the system satisfies all functional requirements, customers can

reject the system if the system fails to meet unstated requirements such as stability, performance, and scalability. But in some cases, the user would accept the system even if there is no 100% correctness. For example, the GPS location might be slightly different due to bad signal conditions. But it is not a big obstacle for the system's final objectives.

- **Reliability**

Reliability refers to how a system can perform the required function with the required precision. In the real world, it is hard to build large systems that are 100% correct and reliable. But high reliability is expected by users. If the failure rate is very low and it does not affect the system's objectives, the customer may still consider the system as an incorrect one. In this system, calculating electricity units consumed values and the bill generating according to the units should be highly reliable and should deliver precise results at every time.

- **Efficiency**

Efficiency refers to the number of computing resources and code that are required by the system to perform its core tasks. Suppose the CEB self-care application requires the latest mobile phones, personal computers, Advance microcontrollers, and operating systems to be installed and used. In that case, the application will not be usable by many customers. If it uses more than the minimum system resources at an unacceptable rate, it will greatly impact the user experience. The users can experience eating issues in their phones due to efficiency issues. And since the mobile application is the most used interface of the system, it is critical to have high efficiency. Not only computational-wise, but the interfaces also have to be efficient when doing some task. The responsive web interfaces should be very lightweight in this system, and all calculations are done within the webserver. Therefore no huge process will be done through the web browser. Thus, the system's interfaces, hardware, and algorithms must be optimized to increase efficiency.

- **Integrity**

Since the project CEB self-care deals with electricity bill payments, the data of the customers, such as their names, emails, contact numbers, and highly sensitive data such as credit card

information, must be used to perform the tasks of the system successfully. This information must not be disclosed or accessed by third parties or even the system administrators for improper usage. Integrity plays a significant role in making a system more appealing to the users in the present system. Hence implementing proper security features in the system is very important in making the system more appealing to a more extensive customer base.

- **Portability**

Portability asks the question of how much effort is required to transfer the program from one hardware platform to another. For example, suppose the developers need to upgrade the hardware platform from Node muc(Arduino base) to more capable Rasbery pi. In that case, it must be easy to perform with minimum modification to the existing subsystems and code. Not only that, the developers can expand their business mode if the system is portable. As an example, if the system can be integrated into the hardware of the main trip switch or in the box, the application can be expanded to many manufacturers growing the business more.

- **Flexibility**

Flexibility refers to the effort that is required to modify or implement a new program or code to the existing system. This is extremely useful to the developers because it will be necessary for offering updates and upgrades to the system's existing functionality. The code should be readable and well commented on, and the use of OOP concepts and good documentation is desired. If the initial design is not flexible, future upgrades and changes would be expensive.

- **Usability**

Usability refers to how much it is easier to use the entire system. In the perspective of the CEB Selfcare App project, there are two types of users. The customers and the administration. The customers use the customer dashboard through a web application, and administration also uses the admin dashboard from the same web application. The system must be easy to install, handle, use and maintain by themselves. The user experience must be enhanced. The

web application must be easy to understand, and the dashboard statistics should be meaningful. Less work, the users of the system, have to do is better for usability.

- **Maintainability**

The effort required to discover and correct a system fault is referred to as maintainability. This is a feature that is desirable by both the customers as well as the developers. If the system can be easily maintained with minimal cost in case of a failure or an error is encountered, then it can be named as a highly maintainable product. In this project, debugging and error-correcting play a major role as the functionality of the system directly could affect human lives. Maintenance can be divided into three categories. They are-

- **Corrective Maintenance**–This is a post-release activity which means removing defects in the existing system. The defects could be already identified ones or new defects identified in the maintenance stages.
- **Adaptive Maintenance**–This concerns adjusting the software in the system to adapt to the changes in the execution environment of the system. For example, if the single-board computer is upgraded from Node Mcu to Raspberry Pi 4B, the software must also be upgraded.
- **Perfective Maintenance**–Improving the already perfected qualities of the system.

All three of the following maintenance operations must be simple to do for a system to be maintainable.

- **Testability**

Testability describes how hard the users and especially the developers have to work to test the system to verify if it achieves the desired quality and performance. If the system is assumed to be testable, it must be simple to find the system's specifications. The testing phase of the CEB self-care app hardware system is crucial as it is highly harmful to its users because the device is working with alternating current, and errors may contribute to so many monetary hazards. And thus, being highly testable is a must-have feature in this program. The software should give both users and developers a simple way to test everything without having to suffer any risk of losing money or time.

## III. Unit Testing

Unit testing refers to testing program units in isolation. A program unit is a piece of code such as a function or method f a class invoked from outside the unit that can invoke other program units. Before being integrated with other units, unit testing is done in isolation. There are two significant reasons for testing a unit in a stand-alone manner. They are-

- Errors discovered during testing can be traced back to a specific unit, allowing them to be quickly corrected. Moreover, unit testing removes dependencies on other program units.
- Second, during unit testing, it is desirable to verify that each distinct execution of a program unit produces the expected result

Unit testing has a limited scope. Therefore programmer will need to verify whether or not a code works correctly by performing unit-level testing. It can be done as follows-

1. Execute every line of code. This is desirable because the programmer needs to know what happens when a line of code is executed. In the absence of such basic observations, surprises at a later stage can be expensive.
2. Every predicate in the unit to evaluates them to true and false separately.
3. Observe that the unit performs its intended function and ensure that it contains no known errors.

Unit testing can be done according to two complementary phases. They are –

- **Static Unit Testing**
- **Dynamic Unit Testing**

In static unit testing, the programmer does not execute the unit; instead, the code is examined over all possible behaviors that might arise during the runtime. Static unit testing the code of each unit is validated against the requirements of the unit by reviewing the code. During the review process, potential issues are identified and resolved.In dynamic unit testing, a program unit is actually executed, and its outcomes are observed. Dynamic unit testing means testing the code by actually running it. In practice, partial dynamic unit testing is performed concurrently with static unit testing. In this project, dynamic unit testing was used.

**Dynamic Unit Testing**

Dynamic Testing is a software testing process used to assess the dynamic behavior of the software code. The main aim of dynamic testing is to test the program's behavior in a runtime environment with dynamic variables or variables that are not constant to identify weak areas. The code must be executed to test the dynamic actions. The primary purpose of the Dynamic Tests is to ensure that the program works correctly before and after the installation of the software, ensuring a stable application without significant flaws. Each variable was evaluated individually using different inputs, as shown in the table below. In this case, rather than using a test driver program, unit tests were performed personally due to the complexity of the programs involved in building the system.

There are multiple test modules that were needed to be tested individually to achieve a good system. They are-

- Login into the system and Fill the Customer register form and Get the email verification link
- Very the email by clicking the URL
- Log in to the system by entering correct details to the login form
- Check the device control panel is connected with the smart meter
- Check pop up alerts works if the smart meter is inactive
- Check whether the meter is shown in red when inactive and when active blue color.
- Unit calculation and bill calculation algorithm
- Smtp email server works for email communication
- Device connection with cloud database
- Web application connect with the cloud database
- Payment forms generation

When testing the above sub-modules of the system, multiple test cases were documented and implemented, covering multiple scenarios where the users of the systems could face. When preparing test cases, it is very important to cover all the methods and backgrounds where an error can occur, which could result in a system failure. That is the real purpose of unit testing. Therefore, when writing test cases, always the mind was on how to fail the system. Then after writing the test cases, the system was subjected to these tests, and then the confronted errors were documented. These documented errors can be later used for debugging purposes.

**System Debugging**

For debugging the system after unit testing, both cause elimination and backtracking techniques were used. Here brute force method was not used as it is not efficient and could leave out critical errors undetected.

1. **Cause Elimination**

This includes two main processes named Induction and deduction. At the induction stage, all data relating to the error found by the unit test are collected and organized into documents. This data is then used to evaluate the problems in order to determine the effects of the related failure. Then the part of the code that caused the error can be inferred. Then using the ordered data, the deducted code phrases that could have caused the error can be confirmed or discarded, and the correct root cause of the error can be isolated. Then in the deduction stage, a list of all potential causes and the probability that they will cause failure in the sub-unit is prepared, and it is used to debug the code to avoid such failures in the decreasing order of their probabilities that the error could occur. That means the errors with the highest probability of occurrence were remedied first, and the lowest ones were done last.

2. **Backtracking**

When an error is found in the backtracking process, the position in the code where the error occurred is first noted. From there, the code is traced back to determine the root cause of the error. This technique was used in the testing process of my project to detect more minor errors in small code segments as this is not suitable for large code segments due to increased complexities in them.

## IV. System Integration Testing

After completing the construction of models, all the models must be integrated. Integrating the system and making the system works is not a straightforward task. Because there are numerous interface errors. In the CEB self-care app system, the following are the separate models that I had to integrate.

- Web application
- Smart meter device
- Cloud Database

The path from tested components to construct a deliverable system contains two major testing phases.

- **Integration testing**
- **System testing**

In order to successfully complete the integration testing phase,

- The entire system must be successfully integrated
- All the test cases should be successfully executed
- All the severe and moderate bugs should be resolved entirely and fixed
- The completed system must be retested completely

Black Box Testing and White Box Testing are two approaches to system integration testing. The tester ignores the code and internal processes in black box testing and places its full attention on the system's outputs. Using the system's inputs and outputs, bugs are detected. Testing of the White Box refers to using the system's internal structures and procedures to monitor and find bugs in the system. For integration testing, this method was used in the project as it is more accurate and easier to find and address bugs at the same time. Incremental testing is the primary technique used for system integration testing. In incremental testing, the tests are carried out as a series of test cycles in an incremental manner. A certain number of compatible and well-defined modules or units are incorporated into each test cycle, and the resulting large composite module is then tested. In this test, before going to the next test cycle, all of the errors found are logged and corrected. This process is carried out until all the subunits and modules are obtained from the final end product. Only then can it be subjected to system testing and testing for acceptance. The resulting core module must be self-sustained and stable when a new module is integrated. Self-sustained means that to carry out the intended function, it must consist of all the code needed. Stable means that without any critical failures or errors, the core module must be able to run unlimited hours straight.

## V. FunctionalTesting

The project was subjected to functional testing after integration testing. This is a black-box testing technique. Here, using some random inputs, the operational requirements of the system are put to the test. The complete system was subjected to random functional testing. In functional testing, the web application was used in multiple mobile phones, desktops, tablets, and using those different devices by different people, the tests were carried out. The fully integrated system was subjected to functional testing, and the people were regarded as the customers. The intention of functional testing was to identify the defects in the user interfaces and the core functionalities in all the subsystems that are critical to the project. Here the users were given the freedom to use the software and hardware interfaces in their liking to identify defects and bugs because we as developers cannot predict how the system will be used by the customers when it is published.

## VI. Acceptance testing

Acceptance testing is formal testing conducted to determine whether a system satisfies its acceptance criteria. It assists the customer in deciding whether to accept or reject the system. The customer generally reserves the right to refuse to take delivery of the product if the acceptance test cases do not pass. There are two major categories in acceptance testing.

- **User acceptance testing**
- **Business acceptance testing**

Actual planning and execution of the acceptance tests do not have to be undertaken directly by the customer. Often third-party consulting firms offer their services to do this. Acceptance testing is only one aspect of the contractual fulfillment of the agreement between a supplier and a buyer.

Acceptance criteria are defined on the basis of these multiple facets of quality attributes. The acceptance criteria I choose for the project as a customer are as follows.

1. **Functional correctness and Completeness**

   In a functional correctness test, the user can be asked, “Does the system do what we want to do it”. Therefore to give a proper answer, all the features which are described in the requirements specification must be presented in the system.

2. **Accuracy**

   Accuracy measures the extent to which a computed value stays close to the expected value. Therefore when testing accuracy, the user can be asked a question like, “Does the system provide correct results”. Most real-time working systems need high accuracy. All the sensors should be working with acceptable working accuracy in my system. Otherwise, it will directly affect the Bill generation, and sometimes users may be lost their profits due to this low accuracy of the system.

3. **Data Integrity**

   The data stored in databases should not be changed while doing retrieve, receive, and execution operations later. The requirement of data integrity is included in the acceptance test criteria to uncover design flaws that may result in data corruption. In the CEB self-care app system, the device sends data using the MQTT server protocol. So the data send through home wifi to the cloud database. In this case, an intruder can sniff the packet or have the ability to do attack the system. Therefore data integrity checking mechanism should be there to detect changes in data packets. Even in some cases, bad people can also attack cloud databases and alter the data. Therefore when using third-party services, the service provider should be trustworthy and every time use paid services. On the other hand, some concepts like database encryption methods should also be followed to prevent those issues from happening in the system.

4. **Backup and Recovery**

   Backup and recovery acceptance criteria specify the durability and recoverbilty levels of the software in each hardware platform. The aim of the recovery acceptance criteria is to outline the extent to which data can be recovered after a system crash.The following questions must be answered in specifying the **recoverability acceptance** criteria-

- How much data can be recovered after a crash, and how?
- Is the recovered data expected to be consistent?

The following questions must be answered in specifying the **backup acceptance criteria**:

- How frequently is the backup process initiated?
- How long does the backup process take?
- Is the backup expected to work online or offline with normal operation suspended during backup?
- Does the backup process check if sufficient storage space is available to accommodate all the data?
- Is the backup process fully automated?

**5. Competitive Edge**

The system must provide a considerable advantage over the existing system in the market by providing innovative solutions. The CEB self-care smart meter and the app is low-cost solution for the current manual meter system used in homes. But apart from the economic perspective, the system must provide in terms of accuracy and features that are innovative in nature when compared with the manual method in order to be appealing to the users.

**6. Usability**

The system must be easy to use by all the users who are using the system despite the age and technical expertise. The installation of the system must not be hard, and the interfaces of the system should be simple, attractive, and interactive to the users. It is better if the system is plug-and-play where the users can just plug into the power and use it. And also, whenever help is needed to the user, it must be available through the system or online. And the system should have overrides and workarounds if an error is encountered.

7. **Performance**

The performance of the system should match the agreed requirements. All the system resources must not be used by the program. As an example, the hardware system should be in power even in the power cuts. Not only that, the hardware system must be able to perform its tasks without lagging. Since this system is a real-time system, lagging will be intolerable. Therefore, the algorithms should be optimized to perform the tasks without lagging. The system should not overheat or produce unnecessary noises by inside cooling fans that use to cool the PZEM -004T sensor, 12 V Dc transformer, and Relay module.

8. **Reliability and Availability**

The reliability of the system, especially a system like this, is very important to the users. The users will rely on the CEB self-care system when switching one off the electric equipment. Because though the system, the device control facility is given o the user. So the system should provide accurate and precise results for the users as outputs 100% of the time in order for the users to be reliable on the system. This must be vigorously tested in the user acceptance test process.

9. **Maintainability and Serviceability**

A system that is critical as this must be subjected to continuous updates and new security feature additions in order to protect the users even more. Therefore, easily maintaining and servicing the system is another key feature that must be tested at the user acceptance testing phase of the project.

10. **Robustness**

Robustness refers to the system's capability to find workarounds if an error is encountered to keep the system functionality smooth enough for the users to still get the use out of the system. For example, if the concentration detection feature fails, the other subsystems should run smoothly without intercepting the user experience. The user acceptance test cases should include tests to test whether the system has this capability properly. The following questions must be addressed in specifying the robustness acceptance criteria:

- What are the types of errors from which the system is expected to recover?
- What are the causes, or sources, of the errors so that these can be simulated in a test environment?
- How are the errors initiated or triggered in the real world?
- What types of corrective and recovery actions are required for each type of error?
- What kinds of disasters can strike? What are those scenarios?
- What is an acceptable response to each of these identified scenarios?
- What is the recovery mechanism for each of the scenarios? Is it workable, understood, and accepted?
- How can disaster be simulated in order to test recovery?

## 11. Startup time

The system should not spend a large amount of time to startup and initiate its actions. Performing the initialization stage of the system should be a minimum time-consuming. Because in most cases, when the power goes, the device can power up to a certain number of hours, or the device should be converted to a sleep state until power resumes. When normal grid power resumes, it will automatically wake. Therefore when the power comes backs, the device should quickly respond to it quickly, and again system should be worked in normal condition.

## 12. Scalability

The scalability of a system is defined as its ability to provide acceptable performance as the following quantity increases effectively**.**

- The geographic area of coverage of a system
- System size in terms of the number of elements in the system
- Number of users
- The volume of workload per unit time

To scale up our system in any circumstance, we must answer the following questions while defining the scalability acceptance criteria.

- How many concurrent users is the system expected to handle?
- How many transactions per unit time is the system expected to process?
- How many database records is the system expected to support?
- What is the largest geographic area the system can cover?

**13. Installability and Upgradability**

The system can be correctly installed and upgraded in the customer environment.If the customer wants to uninstall or downgrade the system software, it must be done smoothly. The following are the system installation and upgrade acceptance criteria.

- The document must identify the person to install the system, for example, the end-user or a trained technician from the supplier side.
- Over what range of platforms, configurations, and versions of support software is the installation or up-gradation process expected to work? The hardware and software requirements must be clearly explained in the document.
- Can the installation or up-gradation process change the user's existing environment? If yes, the risks of this change should be clearly documented.
- If the system includes a licensing and registration process, it should work smoothly and should be documented.
- The installation or up-gradation instructions should be complete, correct, and usable.
- The installation or up-gradation process should be verified during system testing.
- There should be zero defects outstanding against a system installation or up-gradation process.

**14. Confidentiality and availability**

The confidentiality acceptance criteria refer to the necessity that data be secured from unauthorized disclosure, whereas the availability acceptance criteria refer to the requirement that data be protected from authorized users experiencing a denial of service (DoS).

- No unauthorized access to the system is permitted, that is, user authentication is performed.
- Files and other data are protected from unauthorized access.
- The system is protected against viruses, worms, and bot attacks.
- Tools are available for detecting attacks.
- There is support against the DoS attack.
- Privacy in communication is achieved by using encryption.
- All the customer data must be stored in a secure place in accordance with the policies of customer rights, such as confidentiality

**15. Documentation**

The quality of the system user's guide must be at a high standard level. Documentation Acceptance criteria are formulated as follows:

- All user documents must be of high quality in terms of correctness, accuracy, readability, and utility. On the other hand, the software quality assurance team should review and approve this document.
- The online help should be reviewed and signed off by the software quality assurance group.

## VII. Acceptance Test Plan

Then after selecting the criteria, an acceptance test plan must be created. The acceptance test plan must include the following fields.

1) Introduction
2) Acceptance test category (For each category)
    a) Operational environment
    b) Test case specification (could have multiple test cases)
        - Test case ID
        - Test case title
        - Test case objective
        - Testing procedure

- Expected outputs

3) Schedule
4) Human resources

These test cases can be divided into two categories; they are **basic test cases** and **complex test cases**. First, the basic simple test cases are executed, and then, the complex test cases are executed. Using the results of these tests, an acceptance test report is created. The acceptance test report mentions the conclusions gained from the testing process. Not only that, it is mentioned whether the system is acceptable or not.

## VIII. References

[1] Jeff Tian , "Software Quality Engineeringn Testing, Quality Assurance, and Quantifiable Improvement" A JOHN WILEY & SONS ,INC,Hoboken ,New Jersy ,Canada , 2005

[2] K Naik ,P Tripathuy , "Software Testing and Quality Assuarance Theory and Practises" A JOHN WILEY & SONS ,INC,Hoboken ,New Jersy ,Canada , 2008