**THE TITLE OF THE PROJECT**

**Graduation Project Report**

**CMSE or CMPE - 405 or 406**

***NOTE: Leave only correct course code🡪 CMSE 405 or 406 or CMPE 405 or 406***

***delete other unsuitable parts & also this red message completely!***

**Team members:**

**Name, Last Name, Student Number**

**Name, Last Name, Student Number**

**Name, Last Name, Student Number**

**Supervisor:**

**Title and Name of the Supervisor**

**Computer Engineering Department**

**Eastern Mediterranean University**

**Term and Year**

**IMPORTANT FORMAT REQUIREMENTS (*delete these following notes from your report*):**

* When writing your final report**:**
  + Use professional diagramming tools for creating your diagrams, such as: **Modelio**, Visual Paradigm, MS Visio, Draw i.o, etc.
  + Use professional UI designing / prototyping tools such as: Adobe XD, Figma, Proto.i.o, or others.
  + Be careful about **Upper / Lower Case** problems in text when writing your report.
  + References should be cited with numbers between brackets in order of appearance ([1], [2], [3] etc.). The reference numbers in brackets ( [1], [2], [3]...) should be placed anywhere in the report text where you wish to reference them. The same reference can be cited more than once in the text with the same reference number. List your reference under the REFERENCES heading of your report in last page (like [1], [2], [3] etc.).
  + Use two-sided justification for all text  in whole report (except figures)
  + Every figure should have a number and title, be placed in center after the figure, (e.g. ***Figure XX****. Figure Title Finish*.)
  + Every Table should have a number and title, be placed left and above on its table (e.g. ***Table XX****. Table Title Finish*.)
  + There should be 1.5 spaces between lines in whole text. Graphical user interface, application

    Description automatically generated
  + Use **Times New Roman** & **10-11 Punto** in whole text.
* **DO NOT USE UNNECCESARY ENTERS OR GIVE UNNECCESARY SPACES .**
* **REMOVE ALL GUIDING COMMENTS GIVEN UNDER EACH HEADINGS IN THIS REPORT.**

# ABSTRACT

An informative abstract should a) state the scope and principal objectives of the design project, b) describe the methods employed, c) summarize the results and d) state the principal conclusions. It should not simply list the topics covered in the Undergraduate Project Report. In preparing the abstract, you should remember that it will be the most widely read portion of the Undergraduate Project Report. The abstract must be able to stand alone as a very short version of the Undergraduate Project Report rather than as a description of it.

**Keywords:** Around 5 keywords should be listed that are related to the subject area. You should confirm these with your supervisor.

**TABLE OF CONTENTS**

[ABSTRACT II](#_Toc478723910)

[Table of Contents III](#_Toc478723911)

[LIST OF FIGURES V](#_Toc478723912)

[LIST OF TABLES VI](#_Toc478723913)

[1. INTRODUCTION 1](#_Toc478723914)

[2. PROJECT PLANNING AND MANAGEMENT 2](#_Toc478723915)

[3. REQUIREMENTS ANALYSIS 3](#_Toc478723916)

[3.1 Functional Requirements 3](#_Toc478723917)

[3.2 Non-Functional Requirements 3](#_Toc478723918)

[3.3 Realistic constraints 3](#_Toc478723919)

[3.4 Ethical issues 3](#_Toc478723920)

[4. DESIGN 4](#_Toc478723921)

[4.1 High level design (architectural) 4](#_Toc478723922)

[4.2 Software design 4](#_Toc478723923)

[5. IMPLEMENTATION 5](#_Toc478723924)

[5.1 Tools, technologies and platforms used 5](#_Toc478723925)

[5.2 Algorithms 5](#_Toc478723926)

[5.3 Standards 5](#_Toc478723927)

[5.4 Detailed description of the implementation (coding) 5](#_Toc478723928)

[6. TESTING 6](#_Toc478723929)

[7. USER GUIDE OF THE SYSTEM 7](#_Toc478723930)

[8. DISCUSSION 8](#_Toc478723931)

[9. CONCLUSION 9](#_Toc478723932)

[10. REFERENCES 10](#_Toc478723933)

[APPENDICES 11](#_Toc478723934)

[A. Instructions for installing the system 11](#_Toc478723935)

[B. Code for the system 11](#_Toc478723936)

[C. Other relevant material 11](#_Toc478723937)

# LIST OF FIGURES

All figures in the text should be listed here automatically. You can easily do it using Word (*delete these guiding notes from your report)*.

* https://www.youtube.com/watch?v=z6NP31Q1w3M
* https://www.google.com/search?q=microsoft+word+automatic+list+of+figures&sxsrf=AJOqlzWnzj57QlGhQFvYYySdRROkapxcjQ:1675178149239&source=lnms&tbm=vid&sa=X&ved=2ahUKEwiWnZT0jPL8AhWUYPEDHQJaDuMQ\_AUoAnoECAUQBA&cshid=1675178158131275&biw=1366&bih=568&dpr=1#fpstate=ive&vld=cid:0acfd85d,vid:82-uUnpfus4

# LIST OF TABLES

All tables in the text should be listed here automatically. You can easily do it using Word.

# SAME AS DEFINED IN LIST OF FIGURES.

# 1. INTRODUCTION

State the problem that is being solved, why there is a need for the project, who will benefit from the project, attempts by others to solve the problem, general information in the problem area etc.

# 2. PROJECT PLANNING AND MANAGEMENT

You will discuss the planning and management issues are here. You already did some of them in your “PROJECT PLANNING & MANAGEMENT” document. The required and applied whole changes during your project life cycle should be indicated here. Updated work distribution (for group projects) and timeline of the project, indicating tasks and their completion (deadline) estimates should be indicated here. Specifically, updated your project plan, the scope of the plan, your Work Packages (WPs) and your finalized Work Breakdown Structure (WBS) by using MS Project tool, updated sequence activities (pre-requisite WPs or tasks, dependent WPs or tasks, estimation duration of each task), latest cost estimation (i.e., using COCOMO or other cost estimation techniques and tools), latest Gantt chart, latest Critical Path Analysis Flow Diagrams (i.e., PERT, CPM), shortening the planned critical path of the project by pruning critical path activities, by "fast tracking" (i.e., performing more activities in parallel), and/or by "crashing the critical path", applying resource levelling (i.e., resource matrices, histograms), risk planning table (defining risks, probability of each risks, effect of each risk and your strategies for each risk, PFMEA), quality planning, are provided in excellent detail. Briefly:

* Project Definition, Aim, Scope, Target Users etc.
* WBS
* Required Resources
* Gantt Chart (put horizontally on one A4 page)
* Project Package Tables (each package table on a new page)
* Risk Analysis Table
* Procurement Tables
* Perform estimation of effort (Man/month), required total time duration, and required number of team members using COCOMO (or other methods are possible).
* CPM (Critical Path Management) analysis by using PERT (defining paths)
* Creating network diagram of the most critic tasks on your WBS
* Calculating probability of successful completion rate for each paths
* Crashing approach,
* PM tools about planning stage of your project (Brainstorming, SWOT analysis, Affinity, TMAP, etc.) etc. other various techniques can be put here.

|  |
| --- |
| **Figure XX. Gantt Chart of the planned project is given below. Paste its JPEG file below.** |
| Table  Description automatically generated with low confidence |

# 3. REQUIREMENTS ANALYSIS

When you define your functional or non-functional requirements, you can use the following specification techniques:

Table

Description automatically generated

## 3.1 Functional Requirements

Give a verbal description of WHAT the system should do. Who are the users/actors? How will they interact with the system? You need to design use-case diagrams (considering include/extend relationships), a use case glossary, written use cases (Use Case Narratives), QFD analysis, Kano diagram, etc.

When writing your FRs, please focus on User-System and System-System interactions:

* + Use a professional diagraming tools such as: **Modelio**, Visual Paradigm, MS Visio, Draw i.o, etc.
  + Use proper technical jargon such as: “shall be able” to or “should be able to”
  + Use verbs when defining each use case, SHORTLY: ***Display, View, Make, Search, Insert, Add, Delete, Upload, Edit, Calculate, Show, Notify,*** etc…
* Give lists of your Functional Requirements
* Create A Use Case Glossary
* Create Use Case Diagram (Extended and Include)
* Create Written Use Case Tables (or Narratives)
* Quality Functional Deployment (QFD) –use to assess your FRs (you can use edrawsoft, Lucid chart, smartdraw, qfdpro, or any other editing PM tool).
* Kano diagram–use to define the importance of each FRs defined.

## 3.2 Non-Functional Requirements

**Non-Functional Requirements Should Be Listed and Specified Using Metrics and Formulas.**

Security: How difficult should it be for people to hack the system? Reliability: How often will the system be allowed to fail or be unavailable? Usability: How easy should it be for users to use the system? Accessibility: Should people with disabilities be able to use the system? What are the requirements for evolution of the system, such as testability, maintainability, extensibility, and scalability?

Some examples: you may focus on the following NFRs (Property / How you can measure it?):

* Time
  + Transactions / sec
  + Response time
  + Time to complete an operation.
* Space
  + Main memory
  + Auxiliary memory
  + (Cache)
* Speed
  + Processed transactions/second.
  + User/event response time
  + Screen refresh time
* Size
  + Mbytes
  + Number of ROM chips
* Usability
  + Training time
  + Number of choices
  + Mouse clicks
  + Number of help frames or notifications, etc.
* Reliability
  + Mean time to failure
  + Downtime probability
  + Failure rate
  + Availability/Probability of unavailability
  + Rate of failure occurrence
* Robustness
  + Time to recovery
  + % of incidents leading to catastrophic failures
  + Data corruption probability after a failure
  + Time to restart after failure
  + Percentage of events causing system failure
* Portability
  + % of non-portable code
  + Number of systems where software can run
  + Percentage of target dependent statements
  + Number of target systems, etc.

## 3.3 Realistic constraints

What other requirements and constraints should the project meet? Economic: can everybody use the system, or do they need a lot of investment? Environmental: does producing/using the product consume a lot of power? Does it produce pollution? Social: Are there social constraints? E.g. are certain segments of society prohibited in its use (persons less than 18 years old cannot use etc.)? Political: Are there political constraints? E.g. in South Cyprus, it would be a bad idea to produce an application that serves as a guide to Universities in the North. Ethical: Are there any ethical constraints that must be observed? E.g. not “borrowing” ideas or code from other projects without acknowledging it etc. Health and safety: can using the product endanger the health and safety of people or society? Manufacturability: Can the product be manufactured using reasonable amount of resources? Are the resources available? Sustainability: Can the product be used over the long term?

## 3.4 Ethical issues

What are the ethical issues related to the project/product? E.g. can people commit crimes using it? Can they engage in unethical behavior using it?

# 4. DESIGN

## 4.1 High level design (architectural)

Give the system architecture diagram showing all relevant system components/modules and external systems (if any). What are the main modules? Explain them. What are their interfaces? How are they connected to each other?

## 4.2 Software design

Give the modular hierarchy diagram discussing modules and their interfaces, indicate database structure and discuss data structures. Describe the main functions/procedures/methods that are used. Use the IEEE standard and UML tools such as Class diagrams, Associations of Classes, Context Diagrams, Entity-class diagrams for static modeling. State transition diagrams, Communication and/or Sequence diagrams for dynamic modeling. If you have a database, design E-R diagrams, Relational Tables, as well as the Physical DB Tables of the database (attributes, types, constraints, etc.). To do these, apply 1NF, 2NF, 3NF normalization techniques. Design Data Flow Diagrams (DFDs) to create well-structured requirements and depict process modelling of your system. You can also use Decision Tables to represent the logic of choice in conditional statements of your system. Briefly, this section will include;

* 1. A General System Architecture Diagram
  2. UML Modelling
     1. Sequence Diagrams
     2. Activity Diagrams
     3. BPMN diagrams
     4. Class diagrams
     5. Package diagrams
     6. Communication diagrams
     7. State diagrams
  3. DFD Diagrams (Context, Level-0, Level-1, etc.)
  4. E-R Diagrams
  5. Relational Tables
  6. Physical Database Tables, and etc.

# 5. IMPLEMENTATION

## 5.1 Tools, technologies and platforms used

Which tools, technologies, platforms, programming languages etc. did you use?

## 5.2 Algorithms

Give a high-level description of the algorithms used for the main functions of the system. You may use pseudo-code. **Do not put your source codes as figures in this section**. Source codes can be put in the APPENDICES B section in the form of text tables (NOT FIGURES, that increases your report size). Put your source code or pseudo-code as tables, see the example below. To define your algorithms or important source code portions you have, you can prepare similar tables as given below. Each line should be numbered. To easily create such a table, you can use EXCEL and copy your code portion to excel, after numbering, then paste here as a table. You can explain your code lines by referring to its row number while explaining a table. Example:

|  |  |
| --- | --- |
| Table 1. Procedure of the DES algorithm used in the system (EXAMPLE SOURCE CODE TABLE). | |
| 1 | Round: |
| 2 | mixer <-- the mixer component with key k\_i |
| 3 | swapper <-- The swapper component part of a single round |
| 4 |  |
| 5 | function encrypt(sequence) |
| 6 | ciphertext = mixer.encrypt(sequence) |
| 7 | ciphertext = swapper.encrypt(ciphertext) |
| 8 | return ciphertext |
| 9 | endfunction |
| 10 |  |
| 11 | function decrypt(sequence) |
| 12 | plaintext = swapper.decrypt(sequence) |
| 13 | plaintext = mixer.decrypt(plaintext) |
| 14 | return plaintext |
| 15 | endfunction |
| 16 | . |
| 17 | . |
| 18 | . |
| 19 | } // your comments |

## 5.3 Standards

What standards are applicable for the project topic? Which ones did you use? (e.g. standards for coding, designing components etc.) Give a list of standards required to be applied in developing your product.

## 5.4 Detailed description of the implementation (coding)

In this section, describe the system you have implemented in detail, with illustrative diagrams, tables, scenarios, etc. Give representative samples of the code you wrote, explaining how it works. Supplement the code with flow diagrams of modules, so that the context in which the code is used becomes obvious.

# 6. QUALITY AND TESTING

## 6.1 Quality Assurance Activities During Project Life Cycle

Quality Assurance (QA) techniques used, such as reviews or inspections, are needed to be define here. Use the following PM tools in any idea generation and decision-making activity during all the project life cycle. For planning QA, you can use these PM tools.

1. Brainstorming
2. Thought Process Map (TMAP)
3. Quality Functional Deployment (QFD) –use to assess your func reqs
4. Kano model–use to separate the importance of your func reqs
5. Affinity diagrams
6. Fishbone diagrams
7. Check sheets
8. Pareto charts
9. Nominal Grouping Technique (NGT)
10. Delphi Technique
11. SWOT analysis
12. Burndown, Scatter, Control Charts, etc.

For example, you can apply the following QA activities in your project.

1. **Create a Quality Metric Tables that includes (a) a number for each quality metric, (b) a name for the quality metric, and (c) a description of the quality metric. See example;**

|  |  |  |
| --- | --- | --- |
| QUALITY METRIC NUMBER | QUALITY METRIC NAME | DESCRIPTION |
| <Enter quality metric number here> | **<Enter quality metric name here>** | **<Enter quality metric description here>** |
| 11 | Customer Satisfaction | The customers should be given the customer satisfaction form. The average customer satisfaction rating should be 3.0 or better on a scale of 5.0. |
| 12 | Average Response Time | Average Response Time should be less than 5ms. |
| <Add rows as necessary> |  |  |

1. **Create Quality Checklist Tables that includes (a) check of quality activity, (c) a description of the quality issue.See example;**

|  |  |
| --- | --- |
| CHECK | ISSUE/TOPIC |
| <Enter check here> | <Enter quality issue or topic here> |
| √ | Send satisfaction form to customers. |
| √ | Run search/insert/update queries in several complexity to calculate Average **Response Time.** |
| <Add rows as necessary> |  |

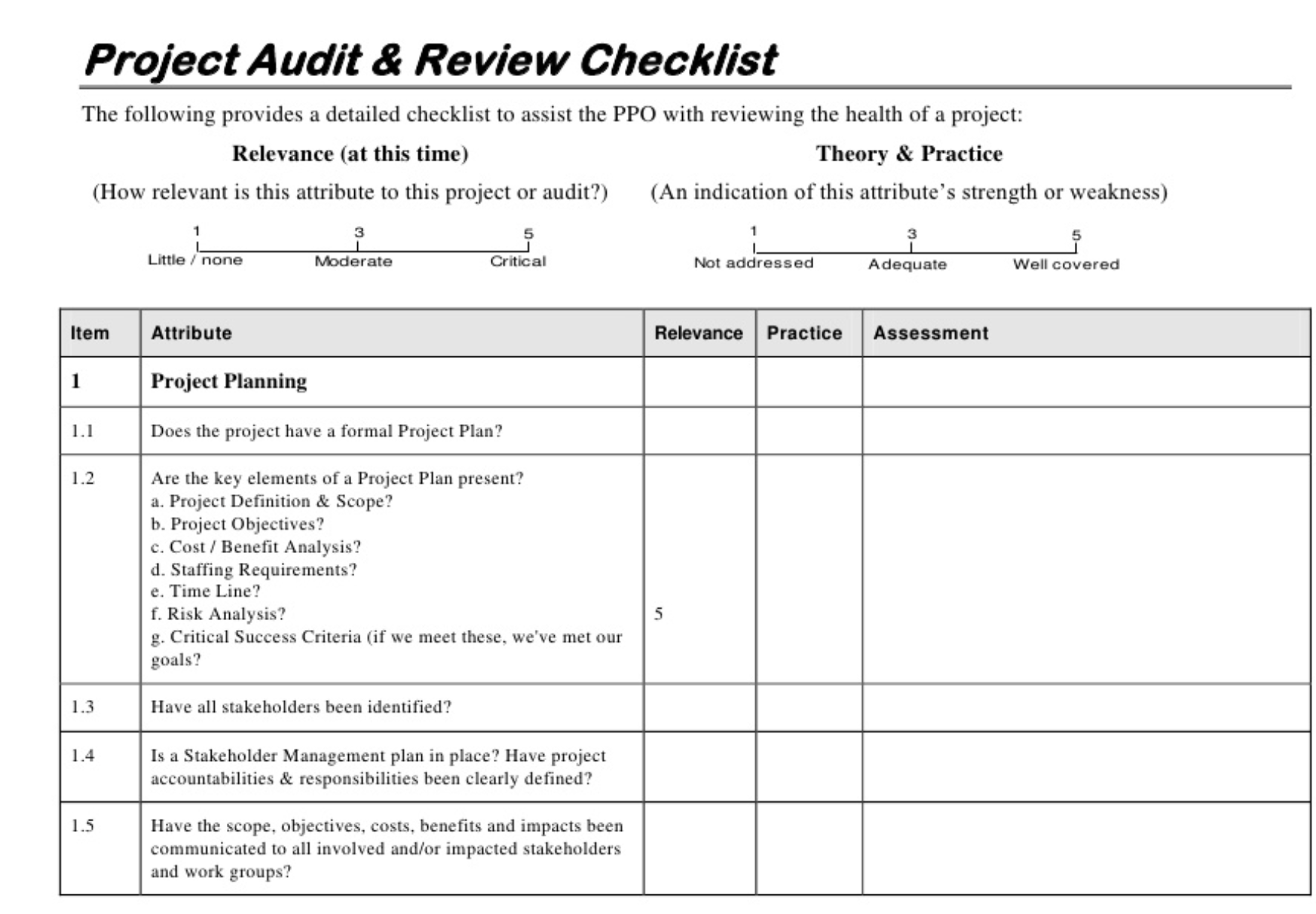
1. **Create Quality Audit Checklist Tables that includes (a) check of quality activity, (c) a description of the quality issue. See examples below;**

|  |  |  |
| --- | --- | --- |
| QUALITY ASSURANCE FUNCTIONS | YES | NO |
| <Enter quality assurance function here> |  |  |
| Assure that all system checks are appropriate and adequate | √ |  |
|  |  |  |
| <Add rows as necessary> |  |  |

**Another Example-1 about Audit plans:**

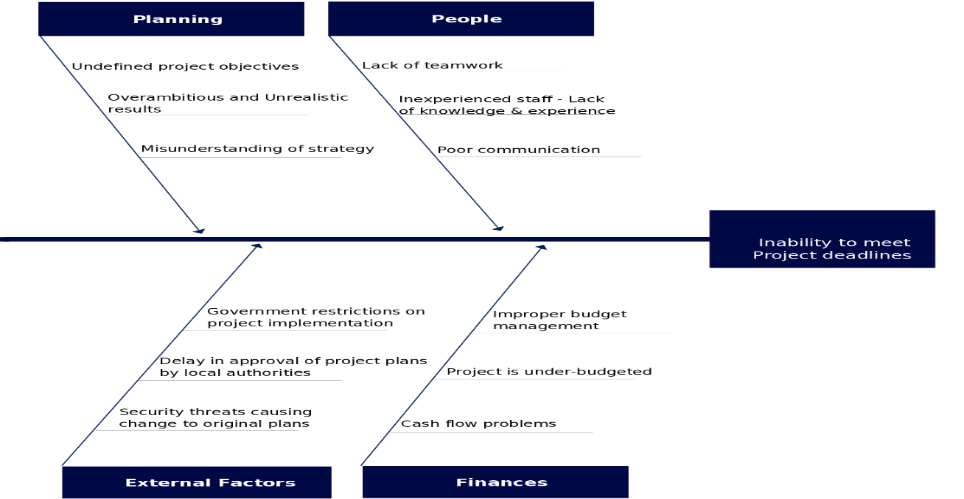
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Audit Date | Planned/  Unplanned Audits | Number of Non-Compliant Issues | | Number of Actions Taken on  Non-Compliant Issues | | # of High-Priority Risk Items in Project Risk List | Hours to Prepare & Apply the Audit | Auditor |
| 5/12 | Planned | 1 | 4 | 0 | 1 | 1 | 12 | GV |
| 6/2 | Planned | 2 | 3 | 0 | 0 | 2 | 24 | GV |
| 3/2 | Unplanned | 2 | 3 | 0 | 0 | 1 | 15 | GV |

**Another Example-2 about Audit plans. You can create a question list for each stages of SDLC:**



**Figure 1**. An audit plan to check the PPM reporting activity.

1. **For QUALITY ASSURANCE; you can apply “Quality Process Analysis (QPA)”. A Cause-and-effect tool can provide an effective summary of critical factors improving your any process.**



**Figure 2**. A Fishbone diagram to assess potential project problems.

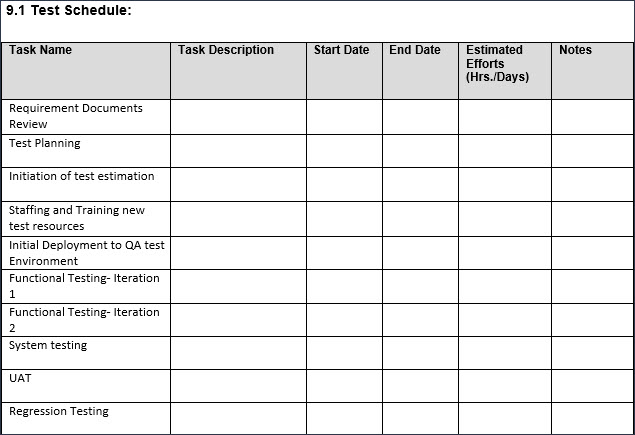
1. **For QUALITY ASSURANCE; you can apply “Statistical Process Analysis (SPA)”. You can use of statistical techniques to control a development or test process or production method is under control or not?**

* R-Chart🡪 <https://www.youtube.com/watch?v=0CtJJGprG6A>
* X-Chart🡪 <https://www.youtube.com/watch?v=RiKUZqW41UM>
* P-Chart🡪 <https://www.youtube.com/watch?v=LSd-1CJQbs8>
* C-Chart🡪 <https://www.youtube.com/watch?v=RvEjgsYkTZk>

## 6.2 Quality Control (QC) Activities After Implementation Completed

How did you test your system? What is your test data? What deficiencies/errors did the test results reveal? What corrections did you make as a result of deficiencies/errors discovered in the testing stage? How did you verify/validate the end product?

1. **You can define a schedule for your test activities. See example below.**



1. **You can perform some inspections and test activities on the following items you produced.**

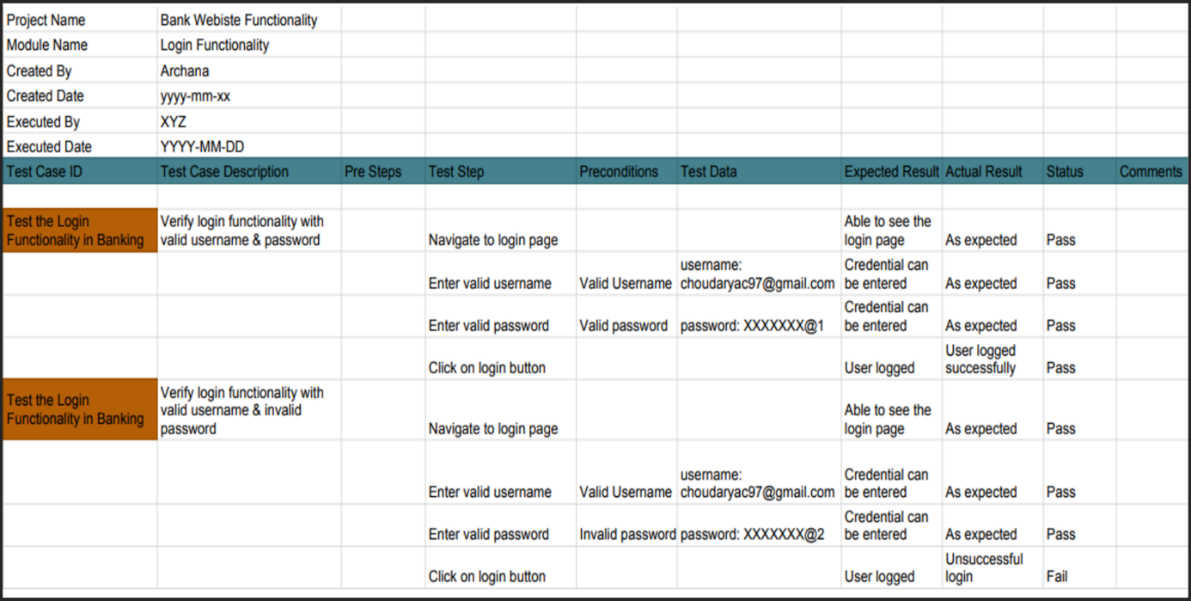
8.2 Inspections Testing.eps

1. **Prepare some Inspection Tables, see examples:**

|  |  |
| --- | --- |
| Fault class | Inspection check |
| Data faults | * Are all program variables initialized before their values are used? * Have all constants been named? * Should the upper bound of arrays be equal to the size of the array or Size -1? * If character strings are used, is a delimiter explicitly assigned? * Is there any possibility of buffer overflow? |
| Control faults | * For each conditional statement, is the condition correct? * Is each loop certain to terminate? * Are compound statements correctly bracketed? * In case statements, are all possible cases accounted for? * If a break is required after each case in case statements, has it been included? |
| Input/output faults | * Are all input variables used? * Are all output variables assigned a value before they are output? * Can unexpected inputs cause corruption? |
| Interface faults | * Do all function and method calls have the correct number of parameters? * Do formal and actual parameter types match? * Are the parameters in the right order? * If components access shared memory, do they have the same model of the shared memory structure? |
| Storage management faults | * If a linked structure is modified, have all links been correctly reassigned? * If dynamic storage is used, has space been allocated correctly? * Is space explicitly deallocated after it is no longer required? |
| Exception management faults | * Have all possible error conditions been taken into account? |

1. **Prepare your Test-Cases to test the critical modules of your system, see example:**

**Use tools:** [**https://docs.optimizory.com/display/rmsis/Test+Case+Operations**](https://docs.optimizory.com/display/rmsis/Test+Case+Operations)



**Main Components of a Test Case:**

* **Test Case ID:** the unique value to identify the TC or sometimes to determine the number of TCs.
* **Module Name:** Module or Function Name which will be tested by TC.
* **TC Scenario:** A functionality of module to test
* **Test Case Name:** Unique name for TC to know what it verifies.
* **Pre-requisite:** any special setup is needed to be done before running TC.
* **Test Data:** the data that needs to be prepared for testing.
* **Test Steps:** describe the steps to perform tests.
* **Expected Results:** expected results from the steps performed above.
* **Actual Result:** usually it will be pass or fail.
* **Status:** After execution, if TC is Pass / Failed or On Hold for some reason
* **Comments:** This column is used to note related information when performing test cases.
* **Test Executed By:** Name of the tester who executed the TC

# 7. USER GUIDE OF THE SYSTEM

Explain to the reader how the system should be used. In addition to verbal description, use screen shots if applicable, or any other illustrations as necessary. Use professional prototyping or programming tools in creating your diagrams, such as: Modelio, Visual Paradigm, MS Visio, Draw i.o, etc.

# 8. DISCUSSION

What is the impact of your solution in the global, economic, environmental and societal context? How will your solution affect the world in general? How will it benefit people/society economically? (i.e. will people/society make/save money through the use of your solution?) Will environment be helped through your solution? (e.g. does it result in energy savings, reduced air pollution, reduced paper usage which reduces the number of trees cut etc.) Will your solution help society? (e.g. by reducing crime rate, helping people get easier access to medical assistance etc.). The answer to this part will be used to assess outcome h of ABET, so write it carefully and thoroughly.

# 9. CONCLUSION

Give a summary of what the project is for, what it does, and why it is a useful project. What have you really achieved through this project? Also, how did it help you personally? What new things you learned? Etc.

# 10. REFERENCES

List your references here. You should refer to them where they are first mentioned in your report ( in text [1], [2], [3] etc.). The same reference can be cited more than once in the text with the same reference number.

1. Reference 1.
2. Reference 2.
3. Reference 3.
4. …

# APPENDICES

## A. Instructions for installing the system

Give step-by-step instructions on how to install the system and get it running. For software projects, all code and other software should be given on a CD as well, labelled by the project name, and the team member names/student numbers. The CD should also contain a soft copy of this report, and a file called “readme.txt” that contains the same instructions as in this section.

## B. Code for the system

Give the code for your project here. If it is too much (more than 10 pages), include only the code for the most important functions. All the code should be on the CD as well, as explained in part A.

## C. Other relevant material

Give other material that is not included in the main body of your report which you think is relevant for your project here.