INDEX

Practical no-1: To identify the role of the software in today’s world across a few significant domains related to day to day life.

The evolving role of software

- Software engineering is the establishment and use of sound engineering procedures in order to obtain economic development software that is reliable, works efficiently on real machines.

- Software engineering is the study and application of engineering to the design, development, and maintenance of software.

- Typical formal definitions of software engineering are: “the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software engineering is relatively a new area of engineering though, but the scope of software engineering is extremely broad. Being one of the prominent branches of the field of Engineering, it’s growing among the fastest fields in the world today.

- It must be noted that the term software development can be used for every type of software development whether it’s as simple as visual basic for applications Modules for Microsoft Word, Excel or Access or developing large, expensive and complicated applications for businesses or creating software for gaming entertainment.

- Software engineers are the computer programming professionals. It’s worth mentioning that a software engineer is also a programmer, as he writes codes, but a programmer may not be called a software engineer, because in the former case, one needs to have a formal education. Besides, a software engineer is the one who follows a systematic process that leads to understanding the requirements, working with teams and various professionals in order to create the application software or components or modules that fulfill the specific needs of the users successfully; whereas a computer programmer can work independently, as he understands algorithms and knows how to create codes following the specifications given by the software engineers. However, software engineering is a vast field. It is not just limited to computer programming, but it’s much more than computer programming. It covers a wide range of professions from business to graphic designs or video game development.

- Not just in a specific field, but every field of work, specific software is needed. Since the software is developed and embedded in the machines in order that it could meet with all intents and purposes of the users belonging to various

- professions, software engineering is of great application and assistance. Not only the field of software engineering involves using some common computer languages, such as, C, C++, Java, python and Visual Basic in an appropriate manner that the intended results may be attained, but it also leads to apply the concepts in such a way that the development of the software may be made effectively and efficiently.
- Software engineers or developers are the creative minds behind computers or programs. Some develop the application software for clients and companies analyzing the needs of the users. Some develop the system software used to run the devices and to control the networks. Whatever be the nature of work, software engineering is one of highest-paid fields in this modern day and age. It's an up-and-coming field, as it's believed that it's likely to grow much faster than the average compared to other professions. If you have strong problem solving skills, an eye for details and good understanding at mathematical functions, then you may consider this lucrative field of study that could give you various benefits including higher level of job satisfaction recompensing your creative efforts.

EDUCATION

- Knowledge of computer programming is a prerequisite for becoming a software engineer. In 2004 the IEEE Computer Society produced the **SWEBOK**, which has been published as ISO/IEC Technical Report 1979:2004, describing the body of knowledge that they recommend to be mastered by a graduate software engineer with four years of experience.Many software engineers enter the profession by obtaining a university degree or training at a vocational school. One standard international curriculum for undergraduate software engineering degrees was defined by the CCSE, and updated in 2004. A number of universities have Software Engineering degree programs; as of 2010, there were 244 Campus Bachelor of Software Engineering programs, 70 Online programs, 230 Masters-level programs, 41 Doctorate-level programs, and 69 Certificate-level programs in the United States.
- In addition to university education, many companies sponsor internships for students wishing to pursue careers in information technology. These internships can introduce the student to interesting real-world tasks that typical software engineers encounter every day. Similar experience can be gained through military service in software engineering.

Airline reservations

- Financial services are not only one of the most dynamic sectors of the economy but also one of the two largest customers of IT. This MSc is jointly run with the University's School of Business in order to give you not only a command of the software technologies that financial institutions require to 'embrace the challenge of change' but also the business context and organisational structures that IT systems need to support.

Banking

- We will concentrate on architectures for building scalable financial software systems, including technologies and techniques that are particularly relevant for the challenges of the financial market - predominantly a need to migrate from mission-critical, monolithic legacy systems to more flexible architectures that allow speedy reaction to customer and business partner’s needs. The technical aspect will be seen in the context of the business environment, where software engineers typically interact with a world of financial jargon and departments with specialised roles and needs.

Games

- Software game is a kind of application that is used not only for entertainment, but also for serious purposes that can be applicable to different domains such as education, business, and health care. Multidisciplinary nature of the game development processes that combine sound, art, control systems, artificial intelligence (AI), and human factors, makes the software game development practice different from traditional software development. However, the underline software engineering techniques help game development to achieve maintainability, flexibility, lower effort and cost, and better design. The purpose of this study is to assesses the state of the art research on the game development software engineering process and highlight areas that need further consideration by researchers. In the study, we used a systematic literature review methodology based on well-known digital libraries. The largest number of studies have been reported in the production phase of the game development software engineering process life cycle, followed by the pre-production phase. By contrast, the post-production phase has received much less research activity than the pre-production and production phases. The results of this study suggest that the game development software engineering process has many aspects that need further attention from researchers; that especially includes the postproduction phase.

Traffic control

- It discusses the main issues that a software engineer must face when building traffic control systems. Thus, this paper will refrain from discussing domain-specific issues (such as control strategies, optimisation algorithms, etc.) about which the authors are not knowledgeable; it will rather concentrate on software engineering. Some of the main issues to be addressed when designing traffic control systems include: - distribution. A traffic control system is, by definition, a distributed one. Acquisition, processing, actuation most likely take place on different nodes. Communication issues must be addressed in a consistent, clean, and, quite often, very efficient way; - complexity of configuration. The "one system fits all" philosophy is clearly not appropriate in such complex systems. However, these systems must be able to easily adapt to changes in configuration (such as a modified street plan) without being rebuilt.

Program 2 : To Identify the problem related to software crisis for a given scenario

In the late 1960s, it became clear that the development of software is different from manufacturing other products. This is because employing more manpower (programmers) later in the software development does not always help speed up the development process. Instead, sometimes it may have negative impacts like delay in achieving the scheduled targets, degradation of software quality, etc. Though software has been an important element of many systems since a long time, developing software within a certain schedule and maintaining its quality is still difficult.

Some scenarios of the software crisis:

1. The Northeast blackout in 2003 has been one of the major power system failures in the history of North America. This blackout involved failure of 100 power plants due to which almost 50 million customers faced power loss that resulted in financia110ss of approximately $6 billion. Later, it was determined that the major reason behind the failure was a software bug in the power monitoring and management system.
2. Year 2000 (Y2K) problem refers to the widespread snags in processing dates after the year 2000. The roots ofY2K problem can be traced back to 1960-80 when developers shortened the 4-digit date format like 1972 to a 2-digit format like 72 because of limited memory. At that time they did not realize that year 2000 will be shortened to 00 which is less than 72. In the 1990s, experts began to realize this major shortcoming in the computer application and then millions were spent to handle this problem.
3. In 1996, Arian-5 space rocket, developed at the cost of $7000 million over a period of 10 years was destroyed within less than a minute after its launch. The crash occurred because there was a software bug in the rocket guidance system.
4. In 1996, one of the largest banks of US credited accounts of nearly 800 customers with approximately $9241acs. Later, it was detected that the problem occurred due to a programming bug in the banking software.
5. During the Gulf War in 1991, the United States of America used Patriot missiles as a defense against Iraqi Scud missiles. However, the Patriot failed to hit the Scud many times. As a result, 28 US soldiers were killed in Dhahran, Saudi Arabia. An inquiry into the incident concluded that a small bug had resulted in the miscalculation of missile path.

Example of the Software crisis:

The production of the os/360 system is a good example of the software crisis. The os/360 was to be produced with the system/ 360 mainframe. Its production started in the 1960 and was planned that by 1966 would be produced. The software was the biggest and most complex having over million lines of code and with an initial investment of 125 million. In Spring 1964 the development task got underway. There were about 70 programmers working on the project but later it was calculated that schedules were slipping therefore they hired more programmers increasing from 60 to 150. But as they increased the number of programmers the less was their standard. Although there was a sudden increase in the number of programmers working on the software, still they estimated that the development was running late by approximately 6 months. Furthermore, a test run was made on the system and was found that the system was very slow which implied that there had to be more reprogramming of already done work which meant more delay in the progress of work. By the end of 1965 it was found out that there were fundamental flaws and there appeared no easy way to arrange them. There was rescheduling in the development plan and it was announced that the software was running 9 months late. At the peak of the system development there was employed a stuff of 1000 people. Finally, by mid 1967 the system was produced a year late of the initial date stipulated and the IBM went with a loss of approximately half a billion. This is a good example of the software crisis when there is a lot of complexity in the system for a number of programmers to produce and when hiring programmers which have a low skill in programming which resulted in a late production of the system and an over budget expenditure which is a loss for the company producing it.

What we infer from this:

Officially known as IBM System/360 Operating System.

A discontinued batch processing operating system developed by IBM

Announced in 1964.

Entered the market in 1967, chock full of errors.

The system was huge, involving more than a million lines of codes, written by hundreds of programmers.

Was the disastrous result of an untried methodology.

Unacceptable due to the tedious performance and the complexity.

Major reasons for software crisis:

- Project running over-budget.
- Project running over-time.
- Software was of low quality.
- Software often did not meet requirements.
- Projects were unmanageable and code is difficult to maintain.

There are some solutions for preventing software crisis:-

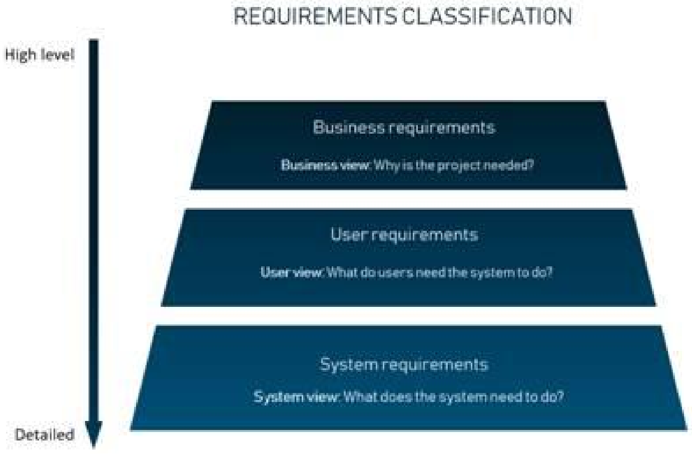
- Experience working as a team member on a software development project.
- Knowledge of basic statics and experimental design.
- Less time and fewer people needed for productive innovation.

PRACTICAL -3

Aim:To classify the requirement into functional & non-functional requirements.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management.

Classification of requirements


REQUIREMENTS CLASSIFICATION
High level
Detailed
Business requirements
Business view: Why is the project needed?
User requirements
User view: What do users need the system to do?
System requirements
System view: What does the system need to do?

High-level requirements cascade down to specific details

Business requirements. These include high-level statements of goals, objectives, and needs.

Stakeholder requirements.The needs of discrete stakeholder groups are also specified to define what they expect from a particular solution.

Solution requirements. Solution requirements describe the characteristics that a product must have to meet the needs of the stakeholders and the business itself.

Nonfunctional requirements describe the general characteristics of a system. They are also known as *quality attributes*.

Functional requirements describe how a product must behave, what its features and functions.

Transition requirements.An additional group of requirements defines what is needed from an organization to successfully move from its current state to its desired state with the new product.

Functional requirements and their specifications

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. Generally, functional requirements describe system behavior under specific conditions. For instance: *As a guest, I want a sofa that I can sleep on overnight*.

Requirements are usually written in text, especially for *Agile-driven projects.* However, they may also be visuals. Here are the most common formats and documents: Software requirements specification document , Use cases ,User stories ,Work Breakdown Structure (WBS) (functional decomposition) ,Prototypes ,Models and diagrams.

Software requirements specification document

Functional and nonfunctional requirements can be formalized in the *requirements specification (SRS)*document. The SRS contains descriptions of functions and capabilities that the product must provide. The document also defines constraints and assumptions. The SRS can be a single document communicating functional requirements or it may accompany other software documentation like user stories and use cases.

SRS must include the following sections:

Purpose. Definitions, system overview, and background.

Overall description. Assumptions, constraints, business rules, and product vision.

Specific requirements. System attributes, functional requirements, database requirements.

It’s essential to make the SRS readable for all stakeholders.

Use cases

Use cases describe the interaction between the system and external users that leads to achieving particular goals.Each use case includes three main elements:

Actors. These are the users outside the system that interact with the system.

System. The system is described by functional requirements that define an intended behavior of the product.

Goals. The purposes of the interaction between the users and the system are outlined as goals.

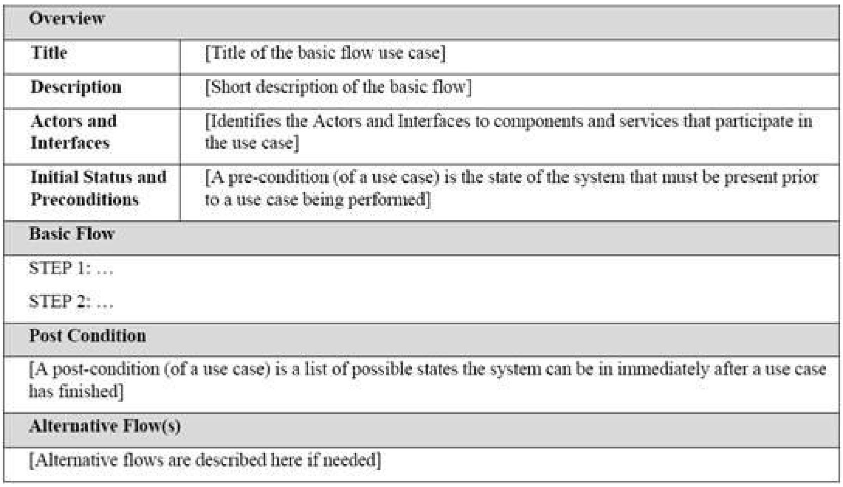
There are two formats to represent use cases:

Use case specification structured in textual format

Use case diagram

A **use case specification** represents the sequence of events along with other information that relates to this use case. A typical use case specification template includes the following information: Description , Pre- and Post-interaction condition , Basic interaction path , Alternative path, Exception path.

*Example:**Use case specification template***



Overview	
Title	[Title of the basic flow use case]
Description	[Short description of the basic flow]
Actors and Interfaces	[Identifies the Actors and Interfaces to components and services that participate in the use case]
Initial Status and Preconditions	[A pre-condition (of a use case) is the state of the system that must be present prior to a use case being performed]
Basic Flow	
STEP 1: … STEP 2: …	
Post Condition	
[A post-condition (of a use case) is a list of possible states the system can be in immediately after a use case has finished]	
Alternative Flow(s)	
[Alternative flows are described here if needed]	

A **use case diagram** doesn’t contain a lot of details. It shows a high-level overview of the relationships between actors, different use cases, and the system.

The use case diagram includes the following main elements:

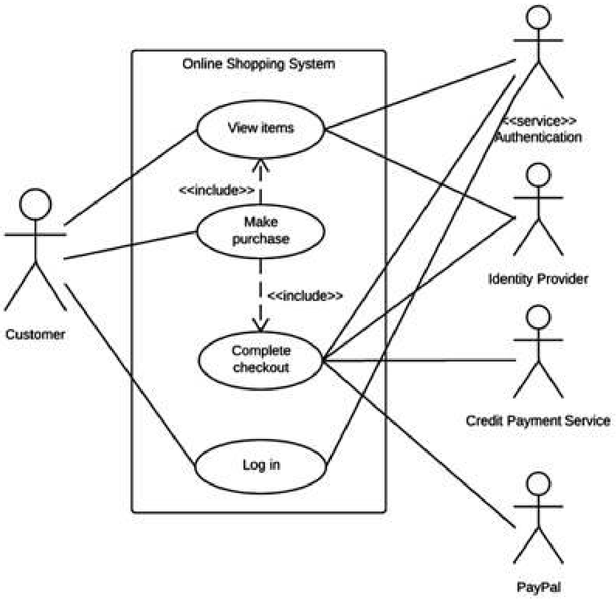
Use cases. Usually drawn with ovals, use cases represent different use scenarios that actors might have with the system (*log in, make a purchase, view items,* etc.)

System boundaries. Boundaries are outlined by the box that groups various use cases in a system.

Actors.These are the figures that depict external users (people or systems) that interact with the system.

Associations.Associations are drawn with lines showing different types of relationships between actors and use cases.

Example:Use case diagram example



User stories-A user story is a documented description of a software feature seen from the end-user perspective. The user story describes what exactly the user wants the system to do. In Agile projects, user stories are organized in a *backlog*, which is an ordered list of product functions.

Example:

As an admin, I want to add descriptions to products so that users can later view these descriptions and compare the products.

User stories must be accompanied byacceptance criteria. These are the conditions that the product must satisfy to be accepted by a user, stakeholders, or a product owner. Each user story must have at least one acceptance criterion.

Finally, all user stories must fit the **INVEST quality model**:

Independent-you can schedule and implement each user story separately. This is very helpful if you implement continuous integration processes.

Negotiable- all parties agree to prioritize negotiations over specification. Details will be created constantly during development.

Valuable-A story must be valuable to the customer. You should ask yourself from the customer's perspective "why" you need to implement a given feature.

Estimatable.A quality user story can be estimated. This will help a team schedule and prioritize the implementation. The bigger the story is, the harder it is to estimate it.

Small. Good user stories tend to be small enough to plan for short production releases. Small stories allow for more specific estimates.

Testable. If a story can be tested, it's clear enough and good enough. Tested stories mean that requirements are done and ready for use.

Nonfunctional requirements

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system's *quality attributes*.

Usability-Usability defines how difficult it will be for a user to learn and operate the system. Usability can be assessed from different points of view:

Efficiency of use:the average time it takes to accomplish a user's goals, how many tasks a user can complete without any help, the number of transactions completed without errors, etc.

Intuitiveness:how simple it is to understand the interface, buttons, headings, etc.

Low perceived workload:how many attempts are needed by users to accomplish a particular task.

Security: Security requirements ensure that the software is protected from unauthorized access to the system and its stored data. It considers different levels of authorization and authentication across different users roles.

Reliability: Reliability defines how likely it is for the software to work without failure for a given period of time. Reliability decreases because of bugs in the code, hardware failures, or problems with other system components.

Performance: Performance is a quality attribute that describes the responsiveness of the system to various user interactions with it. Poor performance leads to negative user experience. It also jeopardizes system safety when it’s is overloaded.

Availability: Availability is gauged by the period of time that the system’s functionality and services are available for use with all operations. So, scheduled maintenance periods directly influence this parameter.

Scalability: Scalability requirements describe how the system must grow without negative influence on its performance. This means serving more users, processing more data, and doing more transactions. Scalability has both hardware and software implications.

A functional requirement describes **what** a software system should do, while non-functional requirements place constraints on **how** the system will do so.

An example of a functional requirement would be:

> A system must send an email whenever a certain condition is met (e.g. an order is placed, a customer signs up, etc).

A related non-functional requirement for the system may be:

> Emails should be sent with a latency of no greater than 12 hours from such an activity.

The functional requirement is **describing the behavior of the system** as it relates to the system's functionality. The non-functional requirement **elaborates a performance characteristic** of the system.

Typically non-functional requirements fall into areas such as:

Accessibility ,Capacity, current and forecast,Compliance ,Documentation,Disaster recovery ,Efficiency ,Effectiveness ,Extensibility ,Fault tolerance ,Interoperability ,Maintainability ,Privacy ,Portability ,Quality ,Reliability ,Resilience ,Response time ,Robustness ,Scalability ,Security ,Stability ,Supportability ,Testability.

PRACTICAL -4

Aim: To implement at least four software metrics.

A **software metric** is a measure of some property of a piece of software or its specifications.

A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operands.Halstead's metrics are included in a number of current commercial tools that count software lines of code. By counting the tokens and determining which are operators and which are operands, the following base measures can be collected:

n1 = Number of distinct operators.

n2 = Number of distinct operands.

N1 = Total number of occurrences of operators.

N2 = Total number of occurrences of operands.

In addition to the above, Halstead defines the following:

n1* = Number of potential operators.

n2* = Number of potential operands.

Halstead refers to n1* and n2* as the minimum possible number of operators and operands for a module and a program respectively.

Halstead metrics :

· **Halstead Program Length –**The total number of operator occurrences and the total number of operand occurrences.

$$N = N1 + N2$$

And estimated program length is, $\hat{N}$= n1$\log_2$n1 + n2$\log_2$

· **Halstead Vocabulary –**The total number of unique operator and unique operand occurrences.

$$n = n1 + n2$$

· **Program Volume –**Proportional to program size, represents the size, in bits, of space necessary for storing the program.

$$V = Size * (\log_2 vocabulary) = N * log2(n)$$

The unit of measurement of volume is the common unit for size "bits".

$$error = Volume / 3000$$

· **Program Difficulty –**This parameter shows how difficult to handle the program is.

D = (n1 / 2) * (N2 / n2)D = 1 / L

As the volume of an implementation of a program increases, the program level decreases and the difficulty increases.

· **Programming Effort –** Measures the amount of mental activity needed to translate the existing algorithm into implementation in the specified program language.

E = V / L = D * V = Difficulty * Volume

· **Language Level –** Shows the algorithm implementation program language level. For example, it is easier to program in Pascal than in Assembler.

L’ = V / D / D

And estimated program level is L^ =2 * (n2) / (n1)(N2)

· **Intelligence Content –**Determines the amount of intelligence presented (stated) in the program This parameter provides a measurement of program complexity, independently of the program language in which it was implemented.

I = V / D

· **Programming Time –**Shows time (in minutes) needed to translate the existing algorithm into implementation in the specified program language.

T = E / (f * S)

Counting rules for C language –

1.Comments are not considered.
2.The identifier and function declarations are not considered
3.All the variables and constants are considered operands.
4.Global variables used in different modules of the same program are counted as multiple occurrences of the same variable.
5.Local variables with the same name in different functions are counted as unique operands.
6.Functions calls are considered as operators.
7.All looping statements e.g., do {…} while ( ), while ( ) {…}, for ( ) {…}, all control statements e.g., if ( ) {…}, if ( ) {…} else {…}, etc. are considered as operators.
8.In control construct switch ( ) {case:…}, switch as well as all the case statements are considered as operators.
9.The reserve words like return, default, continue, break, sizeof, etc., are considered as operators.
10.All the brackets, commas, and terminators are considered as operators.
11.GOTO is counted as an operator and the label is counted as an operand.
12.The unary and binary occurrence of “+” and “-” are dealt separately. Similarly “*” (multiplication operator) are dealt separately.
13.In the array variables such as “array-name [index]” “array-name” and “index” are considered as operands and [ ] is considered as operator.

14.In the structure variables such as “struct-name, member-name” or “struct-name -> member-name”, struct-name, member-name are taken as operands and ‘.’, ‘->’ are taken as operators. Some names of member elements in different structure variables are counted as unique operands.

15.All the hash directive are ignored.

Example –List out the operators and operands and also calculate the values of software science measures like

```
int sort (int x[ ], int n)
{ int i, j, save, im1;  /*This function sorts array x in ascending order */
  If (n< 2) return 1;
  for (i=2; i< =n; i++)
  {  im1=i-1;
       for (j=1; j< =im1; j++)
      if (x[i] < x[j])
      { Save = x[i];
         x[i] = x[j];
         x[j] = save;
       }
 }
 return 0;
}
```

Explanation -

operators	**occurrences**	**operands**	**occurrences**
int	4	sort	1
()	5	x	7
,	4	n	3
[]	7	i	8
if	2	j	7
<	2	save	3
;	11	im1	3
for	2	2	2
=	6	1	3
-	1	0	1
<=	2	-	-
++	2	-	-
return	2	-	-
{}	3	-	-
n1=14	N1=53	n2=10	N2=38

Therefore,

N = 91

n = 24

V = 417 bits

N^ = 86.45

n2* = 3 (x:array holding integer to be sorted. This is used both as input and output)

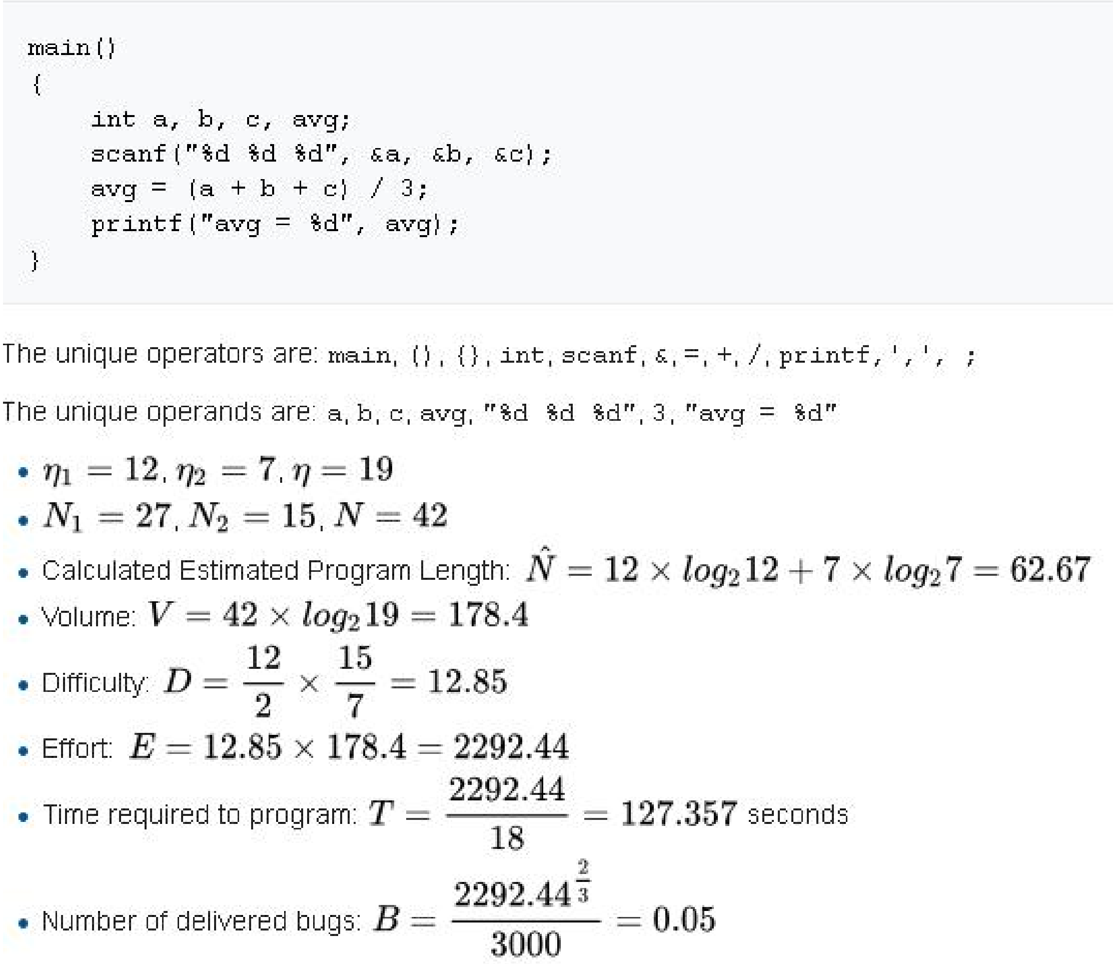
V* = 11.6

L = 0.027

D = 37.03

L^ = 0.038

T = 610 seconds



```
main()
{
    int a, b, c, avg;
    scanf("%d %d %d", &a, &b, &c);
    avg = (a + b + c) / 3;
    printf("avg = %d", avg);
}
```

The unique operators are: main, (), {}, int, scanf, &, =, +, /, printf, ',', ;

The unique operands are: a, b, c, avg, "%d %d %d", 3, "avg = %d"

- $\eta_1 = 12$, $\eta_2 = 7$, $\eta = 19$
- $N_1 = 27$, $N_2 = 15$, $N = 42$
- Calculated Estimated Program Length: $\hat{N} = 12 \times log_2 12 + 7 \times log_2 7 = 62.67$
- Volume: $V = 42 \times log_2 19 = 178.4$
- Difficulty: $D = \frac{12}{2} \times \frac{15}{7} = 12.85$
- Effort: $E = 12.85 \times 178.4 = 2292.44$
- Time required to program: $T = \frac{2292.44}{18} = 127.357$ seconds
- Number of delivered bugs: $B = \frac{2292.44^{\frac{2}{3}}}{3000} = 0.05$

PRACTICAL -5

Aim: Preparation of requirement document for standard application problems in standard format. (e.g Library Management System, Railway Reservation system, Hospital management System, University Admission system)

A Software requirements specification document describes the intended purpose, requirements and nature of a software to be developed. It also includes the yield and cost of the software.

In this document, flight management project is used as an example to explain few points.

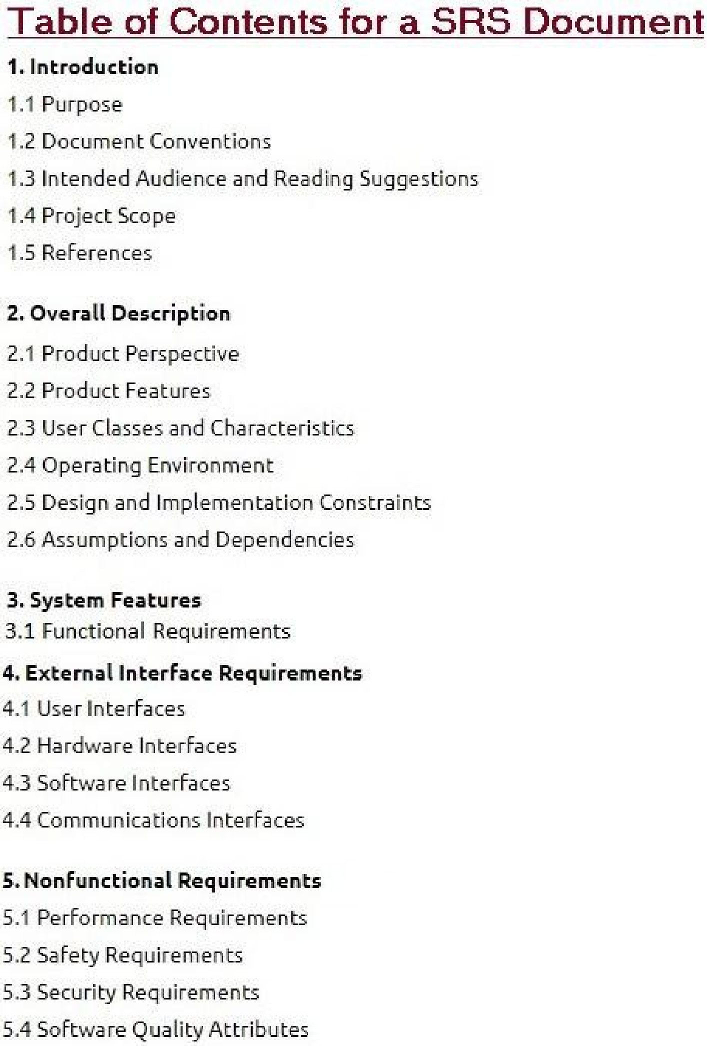


Table of Contents for a SRS Document

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Project Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 Assumptions and Dependencies

3. System Features

3.1 Functional Requirements

4. External Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4.4 Communications Interfaces

5. Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

Student Management System

1st Phase- REQUIREMENT AND ANALYSIS

Problem Definition: Today all the work at the time of admission of the students is done manually by ink and paper, which is very slow and consuming much efforts and time.It is required to Design of a Computerized Automated Student Admission System, to speed up and make it easy to use system.

Purpose:

1). Student Admission System Supports the student admission and registration process, the maintenance of student personal, academic and fee related data.

2). Database maintained by this system usually contains the student's personal, academic and its fee related information. It focuses on storing and processing (insertion, updation) by using web pages

3). Generates student information in formatted html tables, generates the fees invoice.

4). Generate Student's Academic Detail Report.

5). Generate Student's Personal Detail Report.

6). Generate Student's Fee Deposition Status Report.

7). Generate Student's all student's currently deposited their fees.

8). It Stores Merit list provided by University.

Project Requirements:

Ø Automate manual paper work done at the time of student's admission (fees deposition) in the institute.

Ø Eliminate paper work.

ØEfficiently manage the student (academic, personal, fee) details.

Software Required:

Operating System: Microsoft Windows XP

Front End tools: HTML, ASP.NET

Back End tools: MS Access, SQL Server

<u>2nd Phase- DESIGN</u>

The system is a portal used for the automation of the processes associated with the management of final year projects in the department of Electrical and Information Engineering, Covenant University, Nigeria. The processes start from the allocation of project supervisors to students down to the final clearance of the student after the project defence. The processes to be automated are outlined below.

1. Allocation of project supervisors to student.

2. Interface where students can check who their respective supervisors are and the various supervisors being able to check which students have been allocated to them for supervision.

3. Communication Avenue between the students and project supervisor.

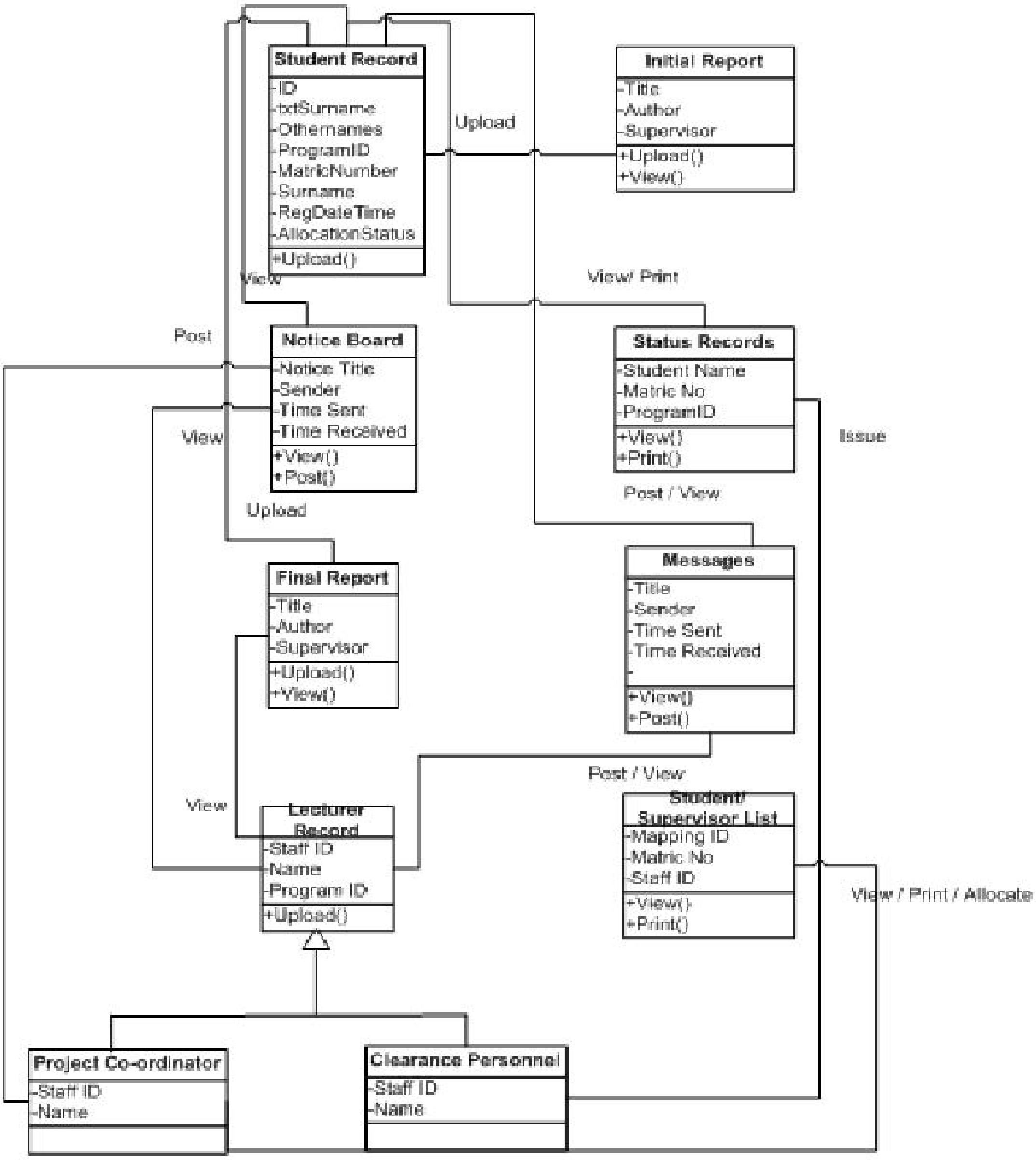
4. Communication Avenue between the student, project supervisor and project co-ordinator.

5. Uploading of the Initial project report to the various supervisors by students.

6. Uploading of final project report to the repository by students.

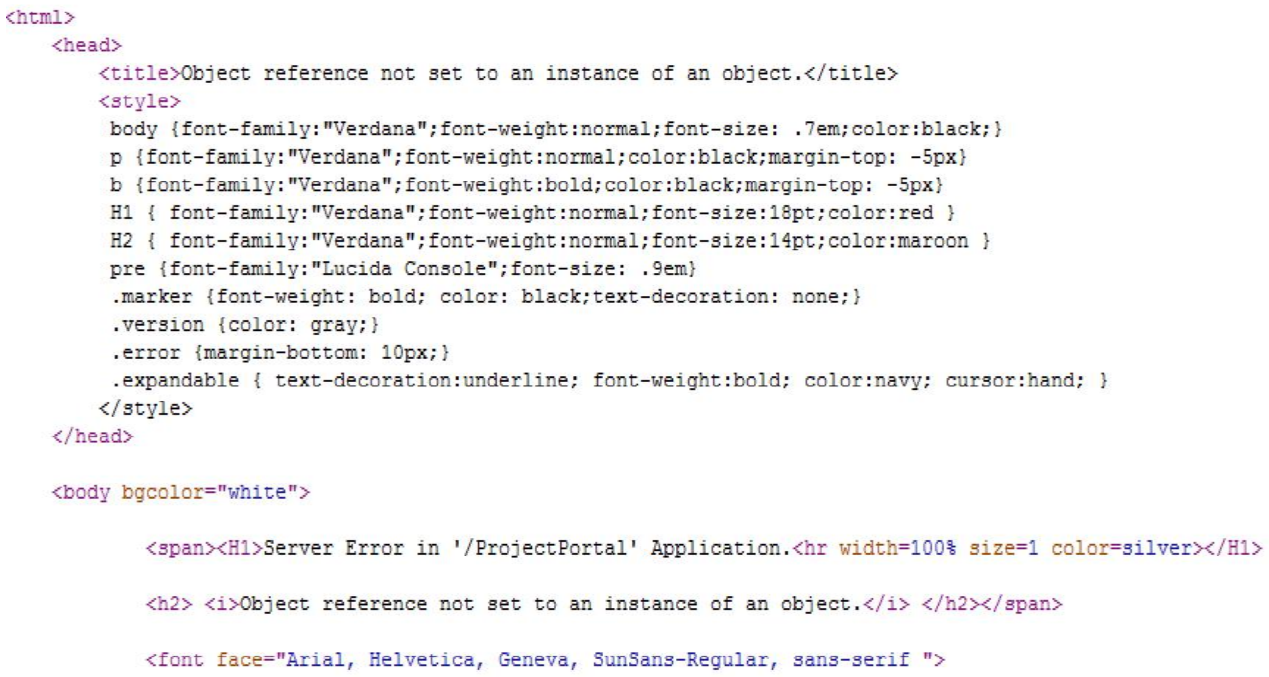
7. Clearance of the students after final year project defence.

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods) and the relationships between the classes. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts. The upper part holds the name of the class, the middle part contains the attributes of the class and the bottom part gives the methods or operations the class can take or undertake.



HYPERTEXT MARKUP LANGUAGE (HTML)

HTML was employed in the front end design of the portal; its tags were used alongside the ones provided by ASP.NET. Cascading style sheets were also employed. Below is a Screenshot of the HTML codes employed in designing a part of the home page.



```
<html>
    <head>
        <title>Object reference not set to an instance of an object.</title>
        <style>
         body {font-family:"Verdana";font-weight:normal;font-size: .7em;color:black;}
         p {font-family:"Verdana";font-weight:normal;color:black;margin-top: -5px}
         b {font-family:"Verdana";font-weight:bold;color:black;margin-top: -5px}
         H1 { font-family:"Verdana";font-weight:normal;font-size:18pt;color:red }
         H2 { font-family:"Verdana";font-weight:normal;font-size:14pt;color:maroon }
         pre {font-family:"Lucida Console";font-size: .9em}
         .marker {font-weight: bold; color: black;text-decoration: none;}
         .version {color: gray;}
         .error {margin-bottom: 10px;}
         .expandable { text-decoration:underline; font-weight:bold; color:navy; cursor:hand; }
        </style>
    </head>

    <body bgcolor="white">

            <span><H1>Server Error in '/ProjectPortal' Application.<hr width=100% size=1 color=silver></H1>

            <h2> <i>Object reference not set to an instance of an object.</i> </h2></span>

            <font face="Arial, Helvetica, Geneva, SunSans-Regular, sans-serif ">
```

3rd Phase- IMPLEMENTATION AND UNIT TESTING

Working of the Present System:-

In present, all work is done manually by hand in bulk of files which is hard to operate and hard to maintain the reports of the student presently, took admission in institute.

1. When the student comes in college.

2. First of all, he/she takes admission form from reception.

3. Fills it and submits it into office.

4. Filled form is first checked with documents like merit list and details came from university and verified by an official person, if there is any mistake then it is corrected.

5. At the time of submission of it the fees is deposited by the candidate.

6. At the time of submission of admission form enrollment no. is assigned to the candidate by the institute.

7. Candidate gets the receipt of fees deposition.

Disadvantages of Present System:-

1. Require much man power i.e. much efforts, much cost and hard to operate and maintain.

2. Since, all the work is done in papers so it is very hard to locate a particular student record when it is required.

Proposed System:-

1. It is automated computerized web based software system.
2. It uses latest technologies like ASP.NET and SQL Server.
3. It is easy to operate.
4. Attractive User Interface

4th Phase- INTEGRATION AND SYSTEM TESTING

Basis path testing, or structured testing, is a white box method for designing test cases. The method analyses the control flow graph of a program to find a set of linearly independent paths of execution. The method normally uses cyclomatic complexity to determine the number of linearly independent paths and then generates test cases for each path thus obtained. Basis path testing guarantees complete branch coverage (all CFG edges), but achieves that without covering all possible CFG paths—the latter is usually too costly. Basis path testing has been widely used and studied.

1. System inputs must be validated.

2. Internal and external inputs conform to formats.

3. Data formats can be mechanically converted into many input data validation tests.

Garbage-In Garbage-Out

“Garbage-In equals Garbage-Out” is one of the worst cop-outs ever invented by the computer industry.

GI-GO does not explain anything except our failure to:

– install good validation checks

– test the system’s tolerance for bad data.

Million Monkey Phenomenon

A million monkeys sit at a million typewriters for a million years and eventually one of them will type Hamlet!

Input validation is the first line of defense against a hostile world.

Input-Tolerance Testing

Good user interface designers design their systems so that it just doesn’t accept garbage. Good testers subject systems to the most creative “garbage” possible. Input-tolerance testing is usually done as part of system testing and usually by independent testers.

Syntax Testing Steps

Identify the target language or format.

Define the syntax of the language, formally, in a notation such as BNF.

Test and Debug the syntax:

– Test the “normal” conditions by covering the BNF syntax graph of the input language.(minimum requirement)

– Test the “garbage” conditions by testing the system against invalid data. (high payoff)

Testing Strategy

Create one error at a time, while keeping all other components of the input string correct. Once a complete set of tests has been specified for single errors, do the same for double errors, then triple, errors, ... Focus on one level at a time and keep the level above and below as correct as you can.

PRACTICAL -6

Aim: To prepare Project Schedule for standard application problems in standard format.

1. PURPOSE

This Project Plan describes what work the **<insert project name>** will do, what results will be achieved, and how project work will be executed and managed. It describes team roles and responsibilities and deliverables. It identifies assumptions, constraints, dependencies, risks, and issues, and it provides high level schedule and budget information.

2. BACKGROUND

This section describes the problem or opportunity the project seeks to address and provides other relevant background information:

·Change in legislation requires action.

·Current technology is outdated and not meeting needs.

·Service levels are low, resulting in frequent customer complaints.

·Demand for products or services are changing.

3. GOAL AND OBJECTIVE

This section lists one project goal and the project objectives. The project goal is a clear, concise statement of the project's purpose and desired results. Project objectives are concise statements of what the project must achieve to realize the project goal. Objectives can be thought of as "sub-goals".

The goal of this project is to reduce traffic accidents. This goal will be achieved if the following three objectives are achieved:

1. Increase public awareness and knowledge of how to drive safely.
2. Pass new, stricter laws for speeding and seatbelt violations.
3. Assign more police to enforce new, stricter laws.

4. SCOPE

This section summarizes the scope of the proposed project by providing a list of key activities and deliverables. In the table provided in this section of the template, list (i) the activities the team will do and (ii) the activities the team will not do but that a reader might mistakenly believe the team is doing.

For example, configuring new software may be in scope for the project team, but training staff on new software may be out of scope because the vendor is providing this service.

5.ASSUMPTIONS AND CONSTRAINTS

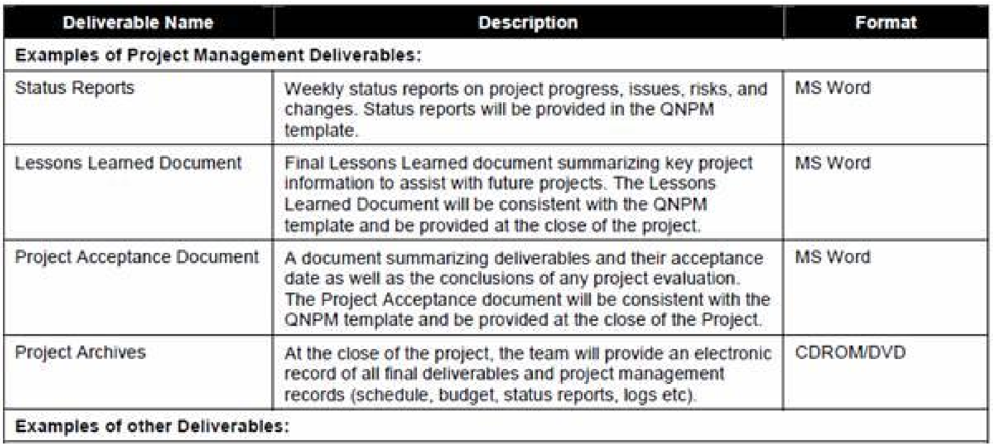
List the project assumptions and constraints in the table provided in the template.

An assumption is a circumstance or event outside the project that can affect its success and that the authors of this plan believe will happen. Constraints are restrictions or boundaries placed upon the project that limit the choices of the project team. The assumptions and constraints for this project are listed in the table below.
Example assumptions: The Sponsor will be available for weekly status meetings and approvals.

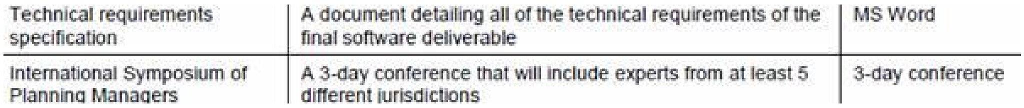
Example constraints:The project must be completed in four months.

6. DELIVERABLES

In the table provided in the template, list each deliverable name with a brief description. This will provide a shared understanding of what is being produced by the team. Deliverables are tangible items that must be produced to complete the project. These can include generic project management deliverables – such as weekly status reports – as well as items specific to the project.
The project is completed when the deliverables listed in the following table are completed.



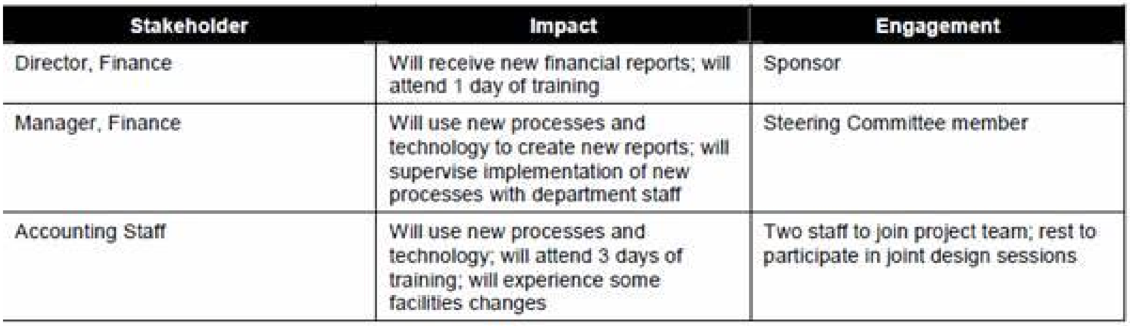
Deliverable Name	Description	Format
Examples of Project Management Deliverables:		
Status Reports	Weekly status reports on project progress, issues, risks, and changes. Status reports will be provided in the QNPM template.	MS Word
Lessons Learned Document	Final Lessons Learned document summarizing key project information to assist with future projects. The Lessons Learned Document will be consistent with the QNPM template and be provided at the close of the project.	MS Word
Project Acceptance Document	A document summarizing deliverables and their acceptance date as well as the conclusions of any project evaluation. The Project Acceptance document will be consistent with the QNPM template and be provided at the close of the Project.	MS Word
Project Archives	At the close of the project, the team will provide an electronic record of all final deliverables and project management records (schedule, budget, status reports, logs etc).	CDROM/DVD
Examples of other Deliverables:		



Technical requirements specification	A document detailing all of the technical requirements of the final software deliverable	MS Word
International Symposium of Planning Managers	A 3-day conference that will include experts from at least 5 different jurisdictions	3-day conference

7. STAKEHOLDERS

This section lists project stakeholders, or people with an interest in or influence over project work and results.It also indicates how stakeholders will be impacted on and engaged by the project.
Examples of key stakeholders are as follows:
·Decision-makers: People with authority and decision-making power over the project.
·Influencers: People who influence and advise decision-makers.
·End users: People who will use the end product of the project.
This section lists project stakeholders, or people with an interest in or influence over project work and results. This section also indicates how stakeholders will be impacted by the project and how they will be engaged.



Stakeholder	Impact	Engagement
Director, Finance	Will receive new financial reports; will attend 1 day of training	Sponsor
Manager, Finance	Will use new processes and technology to create new reports; will supervise implementation of new processes with department staff	Steering Committee member
Accounting Staff	Will use new processes and technology; will attend 3 days of training; will experience some facilities changes	Two staff to join project team; rest to participate in joint design sessions

8. OUTCOMES/SUCCESS MEASURES

Provide a list of statements about the impact the project must have on those outside the project to be considered successful and how those impacts will be measured.
The project will be considered successful if the following statements are true:
·The number of traffic accidents is reduced--as indicated by statistics from MOI.
·Program costs are reduced -- as indicated by actual costs accrued at the end of this fiscal year compared to actual costs accrued over the past five fiscal years.
·There are higher levels of customer satisfaction -- as indicated in a customer satisfaction survey.

9. BUDGET SUMMARY

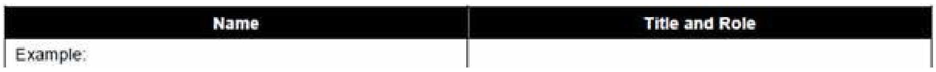
Use the table in the template to provide a summary of the project costs detailed in your Project Budget. Note the budget should be included as an Appendix to the Project Plan. Below the table, list the financial assumptions made during the development of the budget and sources of financial information.
Example assumptions:
·Cost of materials will not increase during this project.
·Salary costs based on 2005 salary levels.
·Consultant costs based on current signed contract.

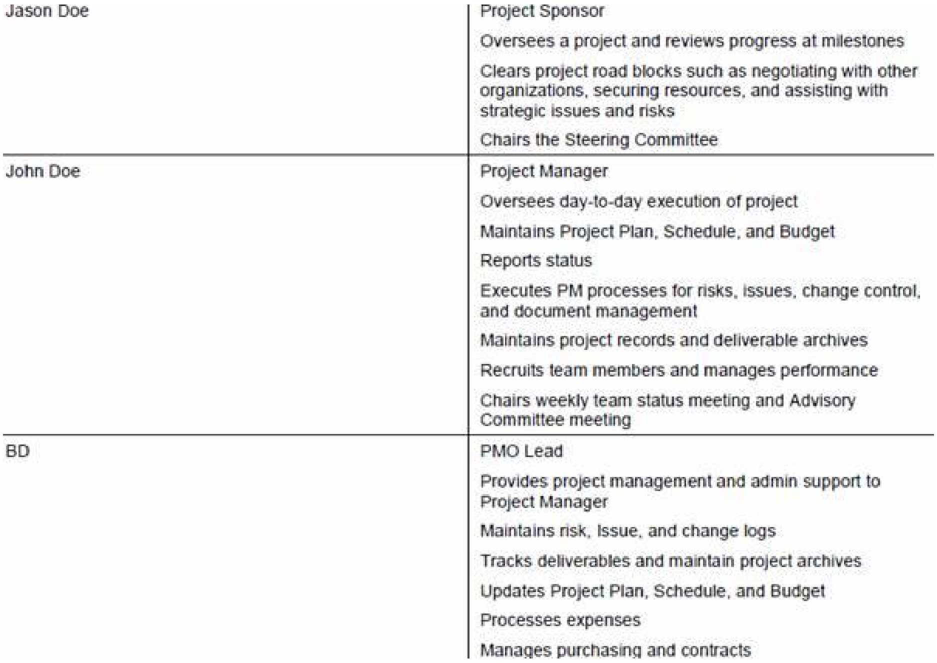
10. HUMAN RESOURCES PLAN

This section contains the results of human resource planning activities completed to support this project. This section describes the roles and responsibilities of the project team, provides their start and finish dates, and provides an organizational chart for the project.
·Start and finish dates for resources must be consistent with the project schedule.
· The names and titles in the roles and responsibilities table, resource schedule, and organization chart should be consistent with each other.
·Remember to include Steering Committee and Advisory Committee members.

10.1 Roles and Responsibilities

Example section: The following resources are required for the project.

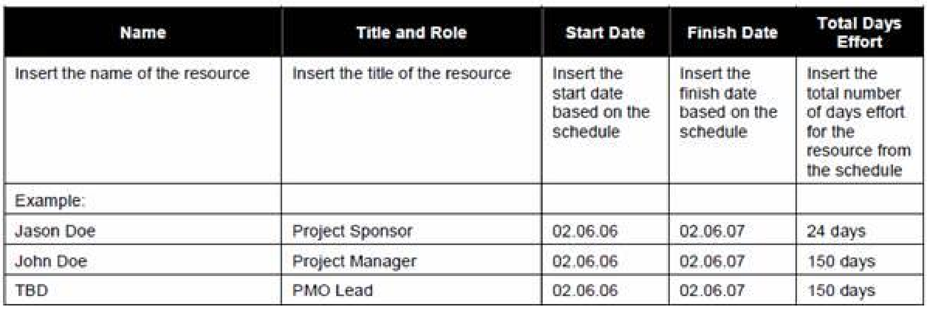




Name	Title and Role
Example:	
Jason Doe	Project Sponsor Oversees a project and reviews progress at milestones Clears project road blocks such as negotiating with other organizations, securing resources, and assisting with strategic issues and risks Chairs the Steering Committee
John Doe	Project Manager Oversees day-to-day execution of project Maintains Project Plan, Schedule, and Budget Reports status Executes PM processes for risks, issues, change control, and document management Maintains project records and deliverable archives Recruits team members and manages performance Chairs weekly team status meeting and Advisory Committee meeting
BD	PMO Lead Provides project management and admin support to Project Manager Maintains risk, Issue, and change logs Tracks deliverables and maintain project archives Updates Project Plan, Schedule, and Budget Processes expenses Manages purchasing and contracts

10.2 Resource Schedule

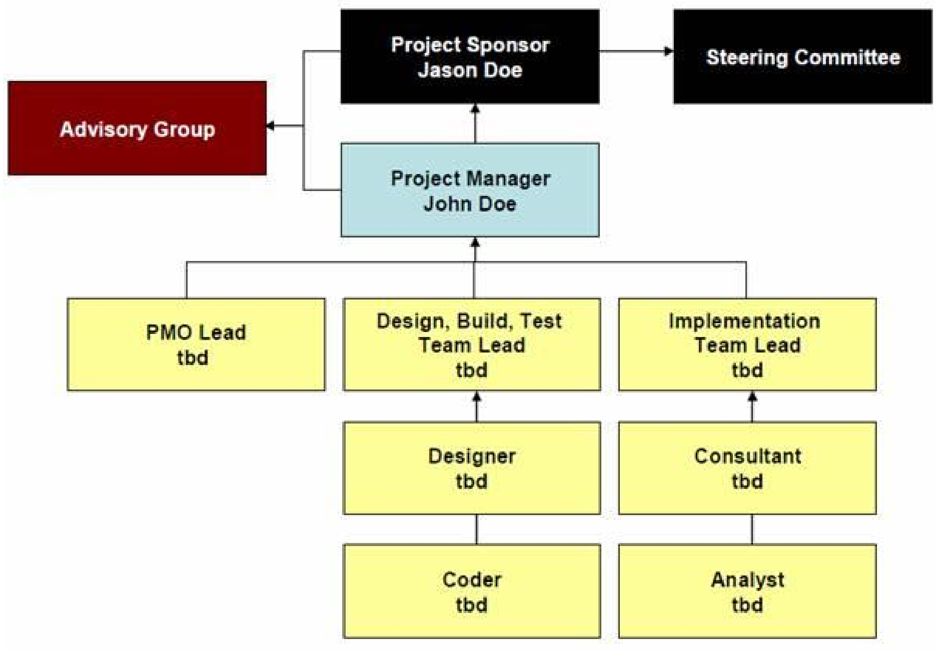
Example section: The following table is a listing of the start and end dates for all resources.



Name	Title and Role	Start Date	Finish Date	Total Days Effort
Insert the name of the resource	Insert the title of the resource	Insert the start date based on the schedule	Insert the finish date based on the schedule	Insert the total number of days effort for the resource from the schedule
Example:				
Jason Doe	Project Sponsor	02.06.06	02.06.07	24 days
John Doe	Project Manager	02.06.06	02.06.07	150 days
TBD	PMO Lead	02.06.06	02.06.07	150 days

10.3 Organizational Chart

Example section: The team will be structured as follows:



11. SCHEDULE SUMMARY

This section lists project milestones and their dates. Milestones are significant dates in a project that typically mark the end of a phase, the completion of a major deliverable, or a major project decision. Milestones are generally used as checkpoints during the project to gauge status and are often used to get approval to continue to the next stage of work.

This project has six milestones. The Project Manager will meet with the Sponsor at each milestone to review progress and obtain approvals and decisions as required.

1. Design complete: October 30, 2006
2. Vendor selected: November 30, 2006
3. Build complete: January 30, 2007
4. Testing complete: February 30, 2007
5. Implementation Plan complete: February 30,2007
6. Implementation complete: June 30, 2007

12. EXTERNAL DEPENDENCIES

This section lists the inbound and outbound external dependencies for the project. “Inbound dependencies” are items the project team requires to continue its work but that are being completed by resources outside of the project team. “Outbound dependencies” are items the project team is producing that others need outside the project.

Example Inbound Dependencies

·Data from Ministry X

·Project Plan – Preparation Guidelines Page 7

·An interface agreement from Ministry Y

·Logo, tagline, and poster from Marketing Firm

·Example Outbound bound Dependencies

·Project Z requires project design documents

·Project Y requires logo, tagline, and poster

13. RISKS

In this section, the project risks identified to date, along with their estimated probability and impact and recommended response are listed. A risk is something that may or may not occur in the future and that can have an impact on the success of the project. “Probability” means the likelihood that a risk will occur and is expressed as High, Medium, or Low. “Impact” describes how seriously the risk could affect the project and is also expressed as High, Medium, or Low.
·Include the risks listed in the Project Definition Document and Business Case if one or both of these was written.
· Low equals <25%
· Medium equals 26%-74%
· High equals 75%- 99%
· Do not include things that have a 100% chance of happening or that have already happened: these are issues, not risks.
· A good risk event statement includes what might happen and its effect on a project.

14. ISSUES

This section lists project issues identified to date and provides a recommended response to the issues. Issues are things that are currently happening and have a negative impact on the project. In the table provided in the template, list any issues that may have been identified while writing the Project Plan along, and include recommended actions to resolve issues.

Only include things that are currently happening or have a 100% chance of occurring in the future.

Sample Project Scheduling that I did was of the project names AR Survial

Project Link :
https://drive.google.com/open?id=1jmxfGPvixxeloAvAzEH9rKdJ652_FleI

It’s a Augmented reality based game that is developed for android Phones
Technologies Used:

UNITY 3D
Android
Photoshop

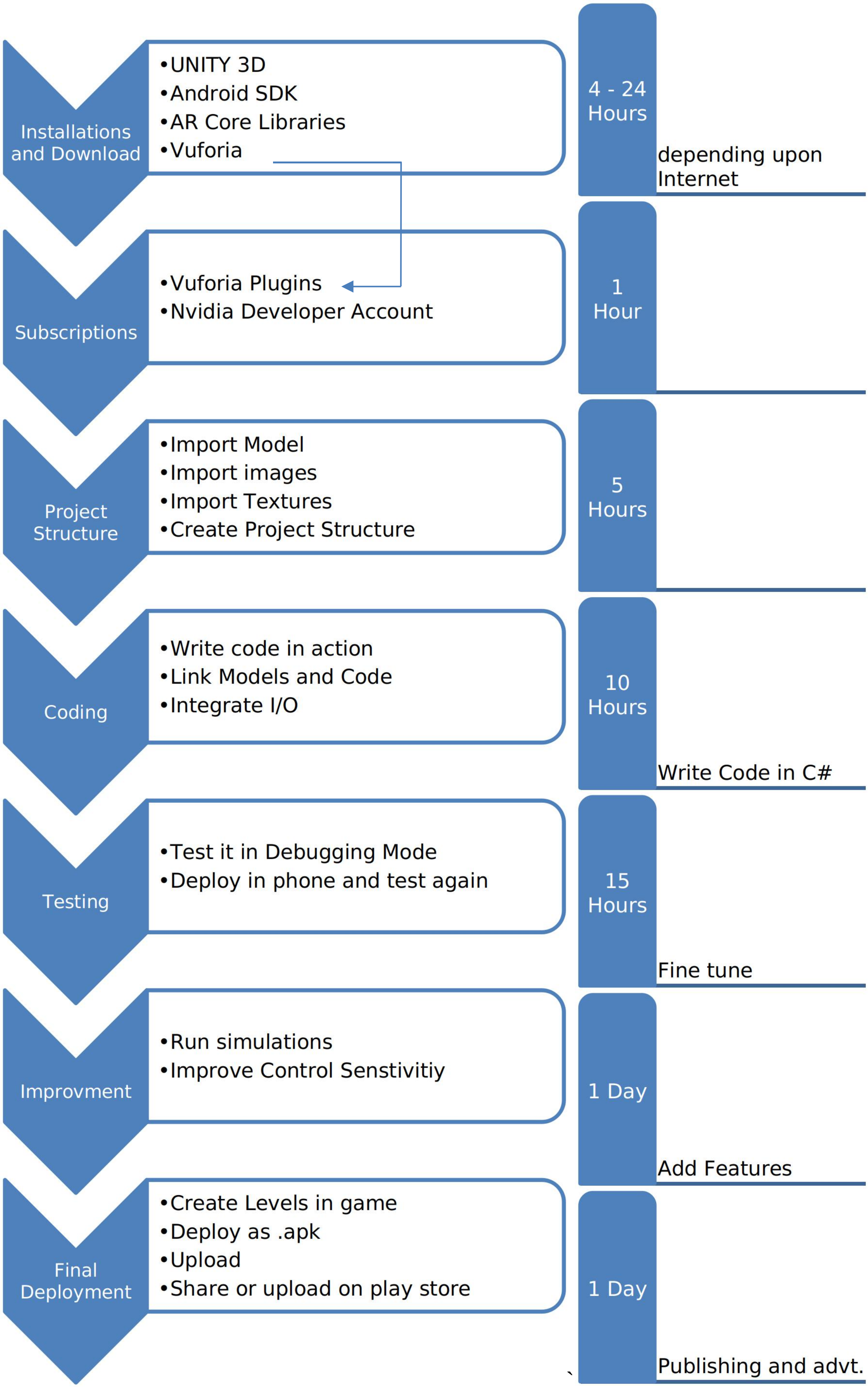
No of team members were 6 including

2 Programmers
2 Designers
1 resource manager and Collector
1 Physics Analyst

Average time given for the project was 3 days.
We classified the tasks into task sets that can run parallel.
And then arranged the task sets in chronological order in which they need to be completed
Installations include UNITY 3D, AR Core libraries, downloading android SDK, downloading models, downloading Plugins. Further Subscriptions was needed for NVidia developer account and some more websites
Then All the stuff that need importing models, textures and images was done Speed of this task set Usually depend upon the speed of processor and HDD or SSD speed
Further the Core task set was done that includes coding and linking models with the physics using C#

All these Steps are useless unless we are able to make the application perform on real grounds
But for that we have to do testing
Testing includes Running App in Debug version and check its memory and CPU usage optimize it

Improvements was done including Custom Sensitivity of control
In final deployment task set we publish our app on play store and then Advt. it so that people came to know about it.



Installations and Download
- UNITY 3D
- Android SDK
- AR Core Libraries
- Vuforia

4 - 24 Hours

depending upon Internet

Subscriptions
- Vuforia Plugins
- Nvidia Developer Account

1 Hour

Project Structure
- Import Model
- Import images
- Import Textures
- Create Project Structure

5 Hours

Coding
- Write code in action
- Link Models and Code
- Integrate I/O

10 Hours

Write Code in C#

Testing
- Test it in Debugging Mode
- Deploy in phone and test again

15 Hours

Fine tune

Improvment
- Run simulations
- Improve Control Senstivitiy

1 Day

Add Features

Final Deployment
- Create Levels in game
- Deploy as .apk
- Upload
- Share or upload on play store

1 Day

Publishing and advt.

PRACTICAL -7

Aim: To implement the functional testing techniques.

Functional testing is a kind of *black box testing* that is performed to confirm that the functionality of an application or system is behaving as expected.It is done to verify all the functionality of an application.There must be something that defines what acceptable behaviour is and what is not.

This is specified in a functional or requirement specification. It is a document that describes what a user is permitted to do so, that he can determine the conformance of the application or system to it.

Therefore, functionality testing can be carried out via **two popular techniques**:

- **Testing based on Requirements:** Contains all the functional specifications which form a basis for all the tests to be conducted.
- **Testing based on Business scenarios:** Contains the information about how the system will be perceived from a business process perspective.

Testing and Quality Assurance are a huge part of the SDLC process. As a tester, we need to be aware of all the types of testing even if we’re not directly involved with them on a daily basis.

Functional Testing Types

Functional testing has many categories and these can be used based on the scenario.

The most prominent types are :

Unit Testing: Unit testing is usually performed by a developer who writes different code units that could be related or unrelated to achieve a particular functionality.This usually entails writing unit tests which would call the methods in each unit and validate those when the required parameters are passed, and its return value is as expected.Code coverage is an important part of unit testing where the test cases need to exist to cover the below three:
1.Line Coverage
2.Code Path Coverage
3.Method Coverage

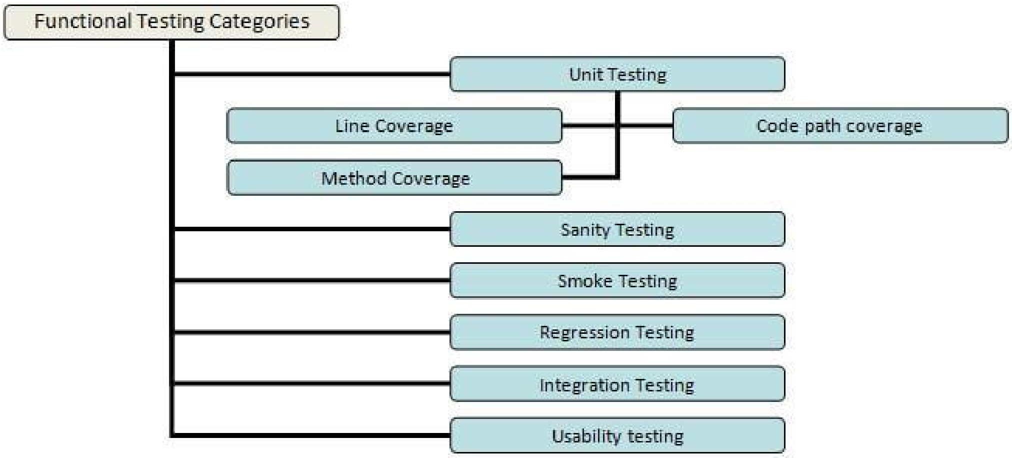
Sanity Testing: Testing that is done to ensure that all the major and vital functionalities of the application/system are working correctly. This is generally done after a smoke test.

Smoke testing: Testing that is done after each build is released to test in order to ensure build stability. It is also called as build verification testing.

Regression tests: Testing performed to ensure that adding new code, enhancements, fixing of bugs is not breaking the existing functionality or causing any instability and still works according to the specifications.Regression tests need not be as extensive as the actual functional tests but should ensure just the amount of coverage to certify that the functionality is stable.

Integration tests: When the system relies on multiple functional modules that might individually work perfectly, but have to work coherently when clubbed together to achieve an end to end scenario, validation of such scenarios is called Integration testing.

Beta/Usability testing: Product is exposed to the actual customer in a production like an environment and they test the product. The user's comfort is derived from this and the feedback is taken. This is similar to that of User Acceptance testing.

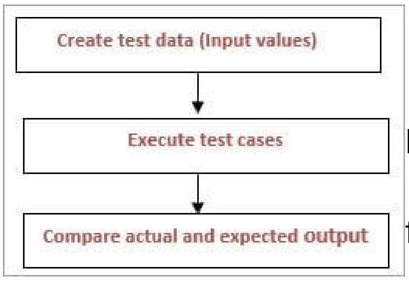


Functional System Testing:

System testing is a testing that is performed on a complete system to verify if it works as expected once all the modules or components are integrated.

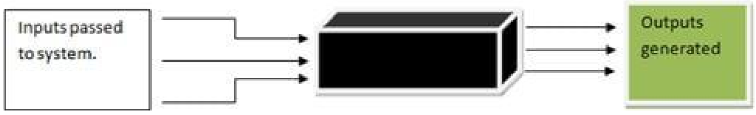
End to end testing is performed to verify the functionality of the product. This testing is performed only when system integration testing is complete including both the functional & non-functional requirements.

Testing process has three main steps:


Create test data (Input values)
Execute test cases
Compare actual and expected output

Functional or behavioral testing generates an output based on the given inputs and determines if the System is functioning correctly as per the specifications.

Hence, the pictorial representation will look as shown below:



Entry/Exit criteria

Entry criteria:

Requirement Specification document is defined and approved.

Test Cases have been prepared.

Test data has been created.

The environment for testing is ready, all the tools that are required are available and ready.

Complete or partial Application is developed and unit tested and is ready for testing.

Exit Criteria:

Execution of all the functional test cases has been completed.

No critical or P1, P2 bugs are open.

Reported bugs have been acknowledged.

Steps Involved

The various steps involved in this testing are mentioned below:

Determine the functionality of the product that needs to be tested and it includes testing the main functionalities, error condition, and messages, usability testing i.e. whether the product is user-friendly or not etc.

Create the input data for the functionality to be tested as per the requirement specification.

From the requirement specification, the output is determined for the functionality under test.

Prepared test cases are executed.

Actual output i.e. the output after executing the test case and expected output (determined from requirement specification) are compared to find whether the functionality is working as expected or not.

Approach

Different kind of scenarios can be thought of and authored in the form of “test cases”. **It mostly has four parts to it:**

Test summary

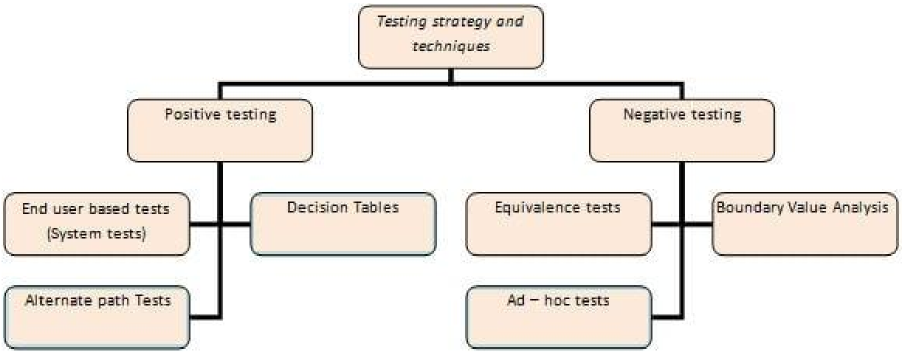
Pre-requisites

Test Steps and

Expected results.

Typically, we would want to uncover the maximum bugs without any escapes with existing tests. Therefore, the QA needs to use optimization techniques and strategize how they would approach the testing.

Functional Testing Techniques



1) End-user based/System Tests

The system under test may have many components which when coupled together achieve the user scenario.In the Example, a customer scenario would include tasks like HRMS application loading, entering the correct credentials, going to the home page, performing some actions and logging out of the system. This particular flow has to work without any errors for a basic business scenario.

2) Equivalence Tests

In Equivalence partitioning, the test data are segregated into various partitions called equivalence data classes. Data in each partition must behave in the same way, therefore only one condition needs to be tested. Similarly, if one condition in a partition doesn't work, then none of the others will work.For Example, in the above scenario the user id field can have a maximum of 10 characters, so entering data > 10 should behave the same way.

3) Boundary Value Tests

Boundary tests imply data limits to the application and validate how it behaves.Therefore, if the inputs are supplied beyond the boundary values, then it is considered to be a negative testing. So a minimum of 6 characters for the user sets the boundary limit. Tests written to have user id < 6 characters are boundary analysis tests.

4) Decision-based Tests

Decision-based tests are centered around the ideology of the possible outcomes of the system when a particular condition is met.

5) Alternate Flow Tests

Alternate path tests are basically run to validate all the possible ways that exist, other than the main flow to accomplish a function.

6) Ad-hoc Tests

When most of the bugs are uncovered through the above techniques, ad-hoc tests are a great way to uncover any discrepancies that are not observed earlier. These are performed with the mindset of breaking the system and see if it responds gracefully.

Advantages of Functional Testing:

This testing reproduces or is a replica of what the actual system is i.e. it is a replica of what the product is in the live environment. Testing is focused on the specifications as per the customer usage i.e. System specifications, Operating system, browsers etc.

It does not work on any if and buts or any assumptions about the structure of the system.

This testing ensures to deliver a high-quality product which meets the customer requirement and makes sure that the customer is satisfied with the end results.

It ensures to deliver a bug-free product which has all the functionalities working as per the customer requirement.

Risk-based testing is done to decrease the chances of any kind of risk in the product.

Disadvantages

There are many chances of performing redundant testing.

Logical errors can be missed out in the product.

This testing is based on the requirement, if in case the requirement is not complete or is complicated or is not clear, performing this testing in such a scenario becomes difficult and can be time-consuming too.

Tools: Sauce Labs, TestComplete, Selenium, SoapUI NG Pro, BrowserStack, Unified Functional Testing, Progress Test Studio

PRACTICAL -8

Aim:To implement the structural testing techniques.

Structural testing, also known as *glass box testing or white box testing* is an approach where the tests are derived from the knowledge of the software's structure or internal implementation.The other names of structural testing includes clear box testing, open box testing, logic driven testing or path driven testing.

Structural Testing Techniques:

· **Statement Coverage -**This technique is aimed at exercising all programming statements with minimal tests.
· **Branch Coverage -**This technique is running a series of tests to ensure that all branches are tested at least once.
· **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch are covered.

Calculating Structural Testing Effectiveness:

Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %
Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %
Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %\

Advantages of Structural Testing:

· Forces test developer to reason carefully about implementation
· Reveals errors in "hidden" code
· Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of Structural Box Testing:

· Expensive as one has to spend both time and money to perform white box testing.
· Every possibility that few lines of code is missed accidentally.

Structural and Functional Technique

Both Structural and Functional Technique is used to ensure adequate testing

Structural analysis basically test the uncover error occur during the coding of the program.

Functional analysis basically test he uncover occur during implementing requirements and design specifications.

Functional testing basically concern about the results but not the processing.

Structural testing is basically concern both the results and also the process.

Structural testing is used in all the phases where design , requirements and algorithm is discussed.

The main objective of the Structural testing to ensure that the functionality is working fine and the product is technically good enough to implement in the real environment.

Functional testing is some times called as black box testing, no need to know about the coding of the program.

Structural testing is some times called as white box testing because knowledge of code is very much essential.

Various Structural Testing are

1.Stress Testing

This testing primarily encompasses the test mechanisms used to validate the system's endurance capacity due to heavy amount of load. Data volume is checked on areas like transactions, disk space, output, memory, CPU utilizations etc. The idea behind stress testing is to determine how far the system's structure is able to handle large volumes of data.

2.Execution Testing

This type of testing aims to detect the performance levels in terms of response and turnaround times. Execution testing checks system's conformance with predefined performance criteria, that is, verifies optimum utilisation of software and hardware. This testing can be performed at any phase of the software development life cycle .

3.Operations Testing

After completing all the previous forms of testing, the product is integrated into the actual operating environment. At this stage, the application is tested under a real environment with normal staff, operations, procedures and documentation. The purpose is to ensure that the requirements are met and are complete and reasonable in the actual operational environment.

4.Recovery Testing

Well, at a certain point of time during testing, it may so happen that the system fails or stops responding. The state of the system is restored and transactions are reprocessed until the point of failure. Recovery testing simply ensures that the system is capable of restoring to the point where it failed. The motive is to preserve data in a secured location.

5.Compliance Testing

This testing helps to verify conformance of the application with the prescribed standards, procedures and guidelines. Compliance testing helps to forecast the success of the project and to enhance the chances of maintainability of the application system. This methodology helps the professionals working on the project comply with the standards so there are chances of least occurrence of errors.

6.Security Testing

Security testing aims at verifying whether integrity of confidential data is strictly maintained. Data protection is the most important part for an organisation as the vulnerability of data loss can hamper the business as a whole

Tools: Testpad, Q test, PractiTest, Zephyr, QMetry, Testrail, Test Collab, QAComplete, TestLink, Squish