2. PROJECT MANAGEMENT.

2.1 PROJECT PLANNING:

Project planning is unquestionably one of the most significant works in developing any project. Before the project can begin, it is best to estimate the work to be done, what resources will be required and how much time will elapse from start to the finish of a project. Planning helped us to prepare a framework that enables us to make a reasonable estimate of all such things.

2.1.1 Project Development Approach and Justification

I have chosen the spiral model of software development for my project. The diagrammatic representation of this model appears like a spiral with many loops. The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study, and the next loop with requirements specification, the next one with design, and so on. Each phase in this model is split into four sectors (or quadrants). The following activities are carried out during each phase of a spiral model.

First quadrant (Objective Setting)

- During the first quadrant, it is needed to identify the objectives of the phase.
- Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk.
- Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

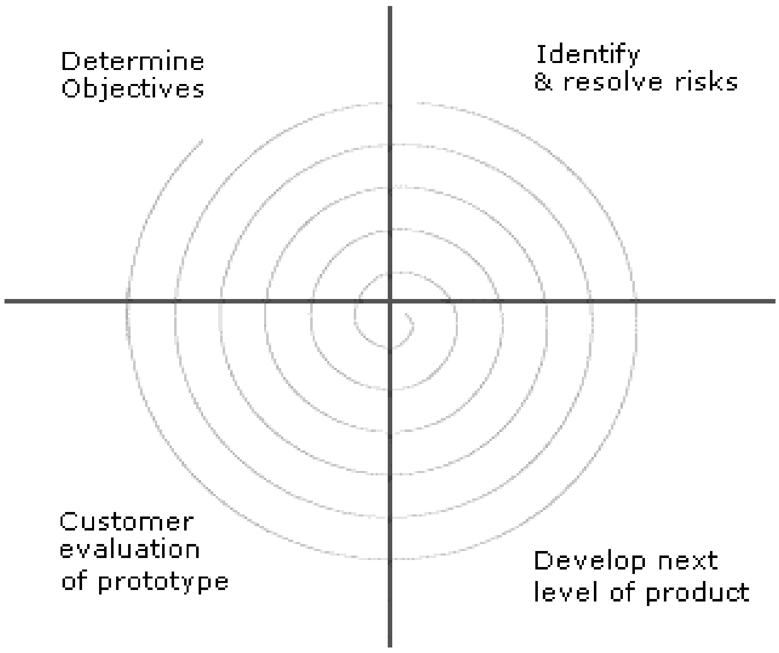
Determine Objectives
Identify & resolve risks
Customer evaluation of prototype
Develop next level of product

Fig 2.1.1 Spiral Model

Third Quadrant (Development and Validation)

- Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.
- Progressively more complete version of the software gets built with each iteration around the spiral.

The spiral model is called a Meta model since it encompasses all other life cycle models. Risk handling is inherently built into this model. The spiral model is suitable for development of technically challenging software products that are prone to several kinds of risks. However, this model is much more complex than the other models.

2.1.2 Project Effort and Time, Cost Estimation

For any software project, it is again equally important to estimate the project effort, time and cost. I have used the concept of Function Point (FP) Analysis and COCOMO Model to implement the same.

Function Point (FP) Analysis

For a software project, the project size is a measure of the problem complexity in terms of the effort and time required to develop the product. Function point (FP) is one the metrics used to estimate this size of the problem. We are interested primarily in estimating the cost and duration of the project. Function point metric was proposed by Albrecht [1983]. This metric overcomes many of the shortcomings of the LOC metric used also as a metric.

One of the important advantages of using the function point metric is that it can be used to easily estimate the size of a software product directly from the problem specification. This is in contrast to the LOC metric, where the size can be accurately determined only after the product has fully been developed.

The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different functions or features it supports. Each function when invoked reads some input data and transforms it to the corresponding output data. A computation of the number of input and the output data values to a system gives some indication of the number of functions supported by the system. Also, in addition to the number of basic functions that a software performs, the size is also dependent on the number of files and the number of interfaces. Function point is computed in two steps. The first step is to compute the unadjusted function point (UFP).

$$\textbf{UFP} = (\text{Number of inputs})*4 + (\text{Number of outputs})*5 + (\text{Number of inquiries})*4 + (\text{Number of files})*10 + (\text{Number of interfaces})*10$$

COCOMO Model

It is said that any software development project can be classified into one of the following three categories based on the development complexity:

- Organic
- Semidetached
- Embedded

Roughly speaking, these three product classes correspond to application, utility and system programs, respectively. My project is a well understood web portal and the size of the development team is extremely small i.e. only one person. Due to all these reasons this software is considered to be **ORGANIC**.

The COCOMO model gives an approximate estimate of the project parameters. The COCOMO estimation model is given by the following expressions:

Effort = a1 x (KLOC)a2 PM

Tdev = b1 x (Effort)b2 Months

Where

- KLOC is the estimated size of the software product expressed in Kilo Lines of Code,
- a1, a2, b1, b2 are constants for each category of software products,
- Tdev is the estimated time to develop the software, expressed in months,
- Effort is the total effort required to develop the software product, expressed in person months (PMs).

Function Point Metrics

Table 2.1.2.1 FP

Sr. No.	Factors	Weights
1.	Does the system require reliable backup and recovery?	**2**
2.	Are specialized data communications required to transfer information to or from the application?	**2**
3.	Is performance critical?	**2**
4.	Does the system require online data entry?	**4**
5.	Does the online data entry require the input transaction to be built over multiple screens or operations?	**1**
6.	Are the inputs, outputs, files, or inquiries complex?	**2**
7.	Is the internal processing complex?	**2**
8.	Is the code designed to be reusable?	**3**
9.	Is the system designed for multiple installations in different organizations?	**0**
10.	Is the application designed to facilitate change and ease of use by the user?	**2**
		Total=20

The Five characteristics for the calculation of FP are:

- No of Inputs:-10
- No of Outputs:-12
- No of Inquires:-9
- No of Files:-5
- No of interfaces:-1

Table 2.1.2.2 Cost Calculation

Parameter	Count(c)	Weighting Factor(W)	W*c
Inputs	10	4	40
Outputs	12	5	60
Inquires	9	4	36
Files	5	10	50
Interfaces	1	10	10
Total			**196**

Once the unadjusted function point (UFP) is computed, the technical complexity factor (TCF) is computed next. TCF refines the UFP measure by considering fourteen other factors. It is given as:

TCF=0.65+0.01*Sum(Weight)

Hence,

FP= Count or Total * [0.65 + 0.01 * Sum (Weight)]

FP= 196*[0.65 + 0.01 *20]

FP=166.6

COCOMO Model

Estimated LOC=2000

Estimated KLOC=2

Now the effort can be calculated using the following basic cocomo equation.

E = ab*(KLOC) ^bb with the parameters of the ORGANIC SOFTWARE PROJECT.

Effort Estimation:

$E = 2.4 \text{ (KLOC)}^{1.05}$ PM

$E = 2.4 (2)^{1.05}$ PM

E = 4.96 ≈**5 PM**

2. Time Estimation:

Tdev = 2.5(Effort (E)) $^{0.38}$

Tdev = 2.5(5)$^{0.38}$

Tdev = **4.6Months**

2.1.3 Roles and Responsibilities

The entire project is handled by only a single person (me) and I am subjected to practice the following roles and responsibilities.

- Requirement Gathering and Analysis
- Transformation of Analysis to Design
- Implementation of the created design on to a programming language i.e.
  PHP for this project
  Mod Security on apache module
  UNIX configuration for setting up a whole WAF.
  Learn about different vulnerabilities by OWASP.
- Testing of the product
- Maintenance of the system

2.1.3.1 Reporting to the Management

The system is regularly monitored by Mr. Mayur Patel (Professor, CHARUSAT) and I meet him regularly for reporting by reviewing the implemented work and taking his advice for carrying out the work ahead and we discuss how the work should be done ahead. We also closely analyze the risks involved, if any, associated with a particular requirement.

Hence, he gives me valuable advice and time and helps me to follow the best way to proceed further.

2.1.4 Group Dependencies

As there is only one person involved in the project, there are no group dependencies

2.2 PROJECT SCHEDULING

Project scheduling consists of describing the order in which the tasks will be carried out or implemented which is in turn described using the Work Breakdown Structure (WBS). Here, it is described as follows:

- Study of similar systems.
- Gathering of requirements
- Generation of Software Requirement Specification (SRS).
- Analysis of gathered requirements.
- Identification of possible risks involved with the requirements.
- Transformation of Analysis to appropriate Design.
- Implementation of own web application firewall setup virtually on BOX.
- Configured proxy server and mod security on apache on the WAF BOX.
- Implementation of requirements using PHP.
- Evaluation of the implemented requirements (a kind of prototype).
- Implementation of complete product.
- Documentation of project work.

2.2.1 PERT Representation

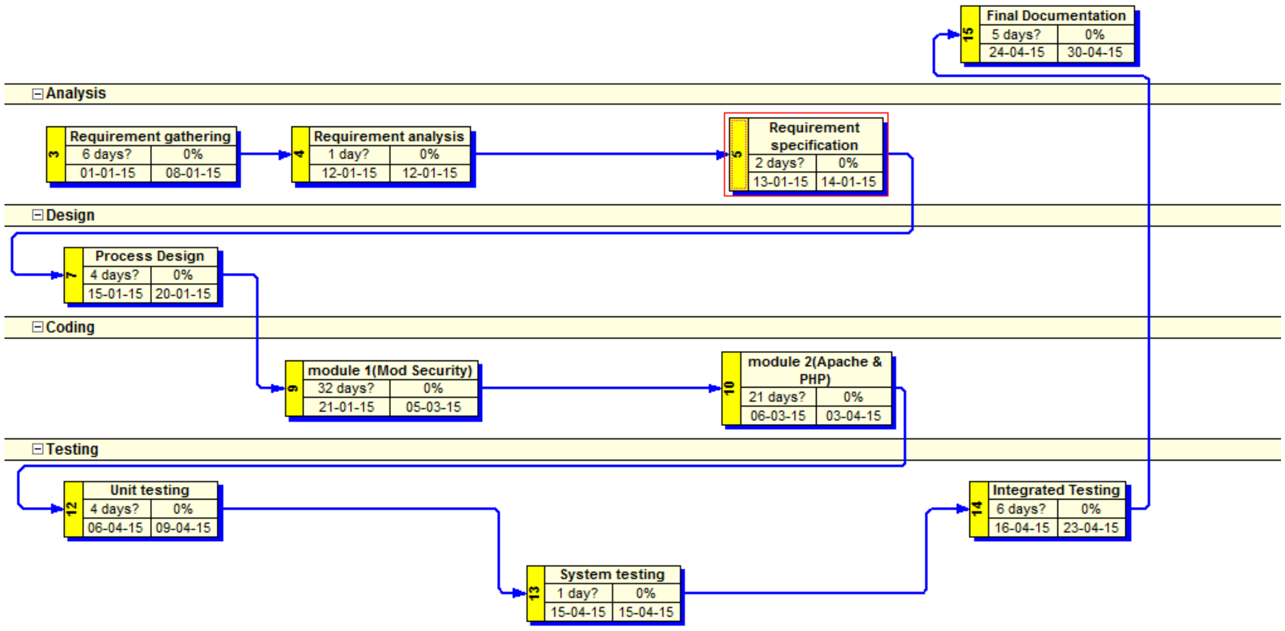

Fig 2.2.1 PERT Representation