Software Project Management Document
Version 1.0
October 12, 2021

"**Apna Classroom**"
Online Classroom

Group 2
Aaryak Shah (2019IMT-001)
Ashes Mondal (2019IMT-021)
Subodh Rajpopat (2019IMT-103)

Submitted in partial fulfillment
Of the requirements of
ITIT-3103 Software Engineering

1. Size Estimation Using Function Point Metric

1.1. Estimating variables

EI (External Inputs) **Total: 7**

a. User Auth Credentials (simple, 3)
b. Classroom Details (average, 4)
c. Announcement Details (simple, 3)
d. Assignment Details (complex, 6)
e. Submission Details (average, 4)
f. Submission Marks (simple, 3)
g. Post Comment (simple, 3)

EO (External Outputs) **Total: 6**

a. Show All Classrooms (simple, 4)
b. Individual Classroom (simple, 4)
c. Posts Feed (average, 5)
d. Individual Assignment (average, 5)
e. To Do List (simple, 4)
f. Results (simple, 4)

EQ (External Inquiries) **Total: 1**

a. Automated Email Notifier (simple, 3)

ILF (Internal Logical Files) **Total: 2**

a. ApnaClassroom IIITM Database (complex, 15)
b. Configuration Files (simple, 7)

EIF (External Interface Files) **Total: 0**

N/A

1.2. Unadjusted Functional Point (UFP):

`UFP = 4 * 7 + 5 * 6 + 4 * 1 + 10 * 2 + 10 * 0 =` **82**

1.3. Refining The UFP Based On Complexities

Parameter	SIMPLE	AVERAGE	COMPLEX
EI	4	2	1
EO	4	2	0
EQ	1	0	0
ILF	1	0	1
EIF	0	0	0

```
Refined UFP =
  3 * 4 +  4 * 2 +  6 * 1
+ 4 * 4 +  5 * 2 +  7 * 0
+ 3 * 1 +  4 * 0 +  6 * 0
+ 7 * 1 + 10 * 0 + 15 * 1
= 77
```

1.4. Calculating Function Point

Degree of Influence

Assuming average case for DI, we can grade the 14 Complexity Adjustment Factors at 4

`DI = 14 * 4 =` **56**

Technical Complexity Factor

`TCF = 0.65 + 0.01 * 56 =` **1.21**

Function Point

`FP = TCF * UFP = 1.21 * 77 =` **93.17**

2. COCOMO Estimation of Development Effort and Time

2.1. Lines Of Code Estimation

The majority of the project is written in Javascript, which is a 3GL Language. We can use the estimates table and see that 3GL languages have approximately 80 LOC per function point. However, considering the use of helper frameworks such as React and Express, we can say that the code is more advanced than a standard 3GL. We will assume 30 LOC per function point for this estimate.

$$\text{KSLOC Estimate} = 30 * FP = 30 * 93.17 = 2795 \text{ LOC} = \mathbf{2.79\ KSLOC}$$

2.2. Effort Estimation

We are considering this software to be of Organic Nature. From this we can apply COCOMO as follows:

$$\text{Effort} = 2.4 * (\text{KSLOC})^{1.05} \text{ PM} = 2.4 * (2.79)^{1.05} = \mathbf{7.05\ PM}$$

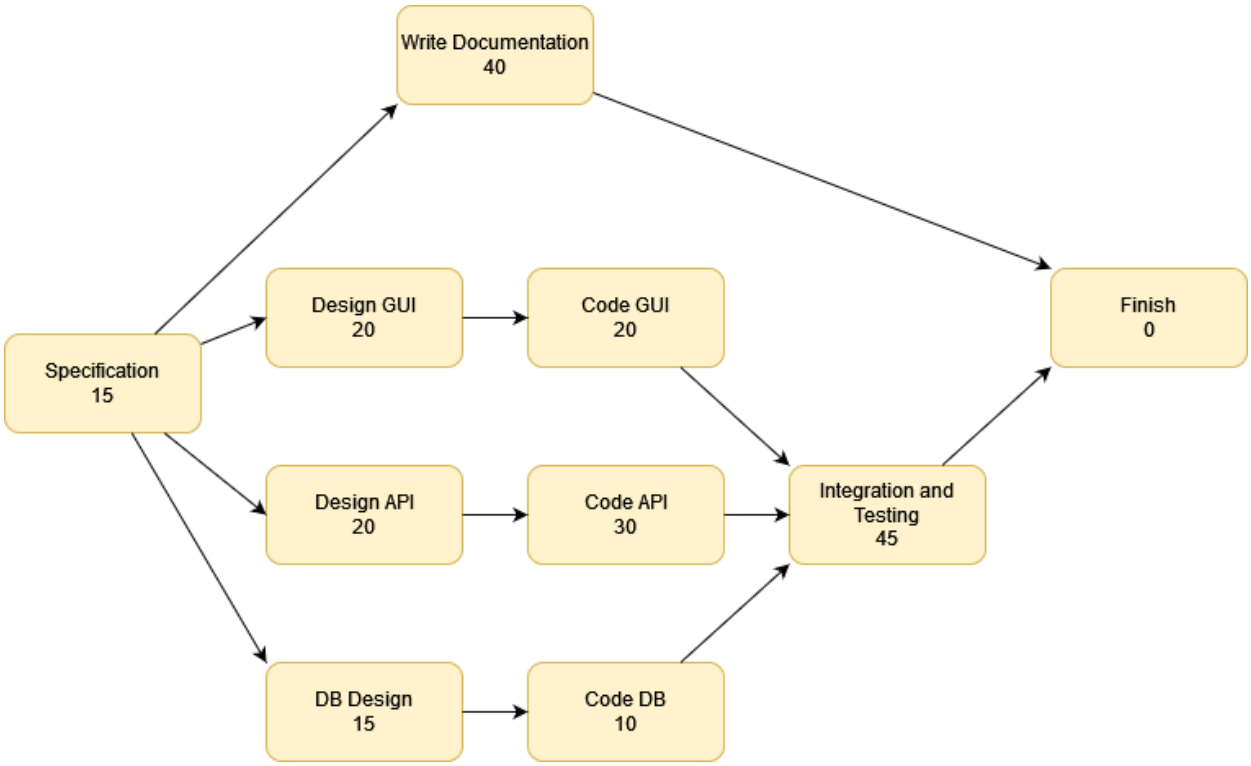
2.3. Development Time Estimate

Given an Organic Project and Effort Estimate of 7.05 PM, we can apply COCOMO as follows:

$$T_{Dev} = 2.5 * (\text{Effort})^{0.38} \text{ Months} = 2.5 * (7.05)^{0.38} = \mathbf{5.25\ Months}$$

3. Scheduling

3.1. Activity Network



Task Number	Task	Duration (Days)	Dependent on Task
T1	Specification	15	-
T2	Write Documentation	40	T1
T3	Design GUI	20	T1
T4	Design API	20	T1
T5	Design DB	15	T1
T6	Code GUI	20	T3
T7	Code API	30	T4
T8	Code DB	10	T5
T9	Integration and Testing	45	T6, T7, T8

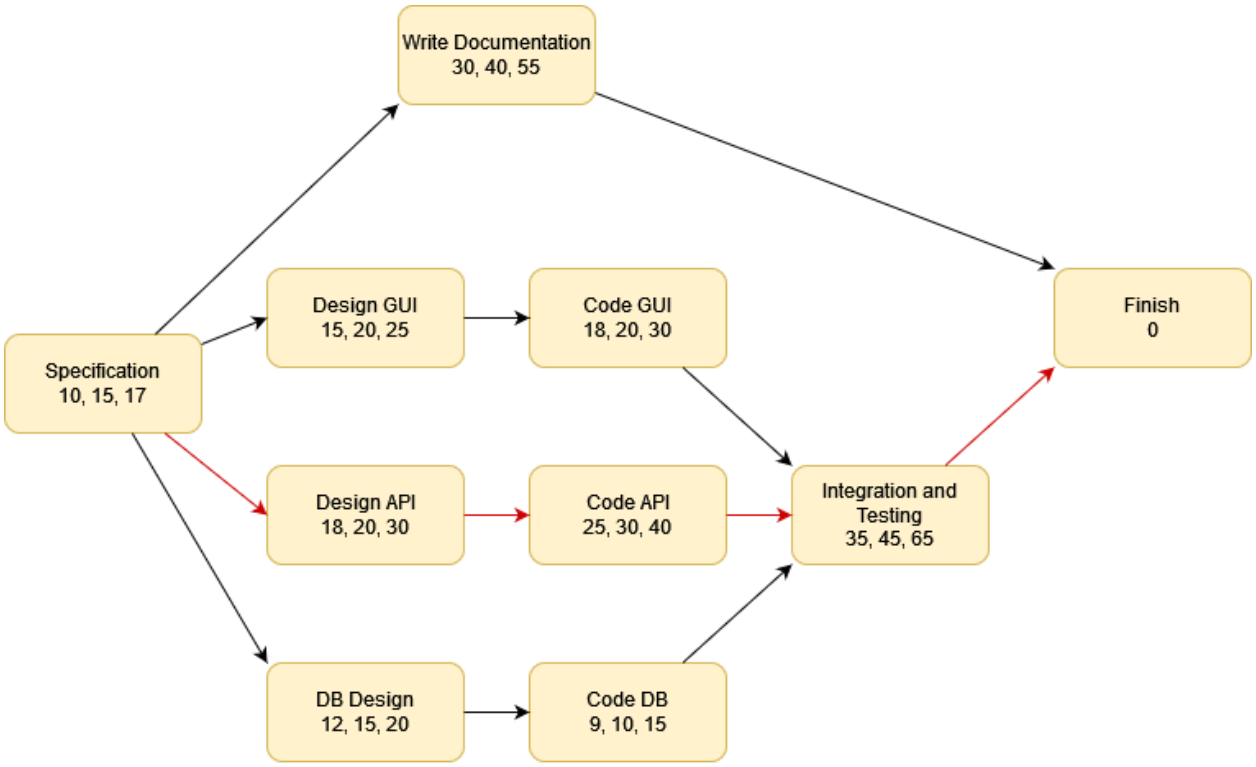
[P.T.O]

3.2. Projected Parameters Computed from Activity Network

Task	Early Start	Early Finish	Latest Start	Latest Finish	Slack Time
Specification	0	15	0	15	0
Write Documentation	15	55	70	110	55
Design GUI	15	35	25	45	10
Design API	15	35	15	35	0
Design DB	15	30	40	55	25
Code GUI	35	55	45	65	10
Code API	35	65	35	65	0
Code DB	30	40	55	65	25
Integration and Testing	65	110	65	110	0

Minimum Time = **110 Days**

3.3. PERT Chart



Critical path is shown in red arrows